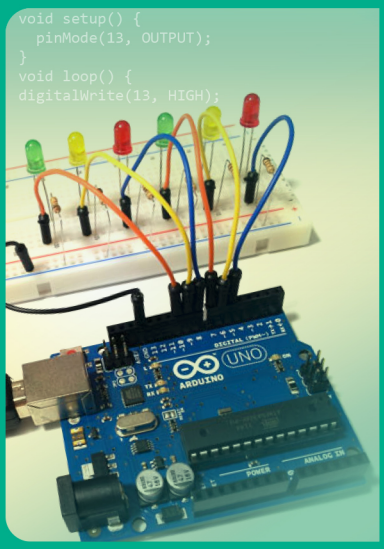


DISEÑO Y CONSTRUCCIÓN DE OBJETOS INTERACTIVOS DIGITALES

Experimentos con la plataforma Arduino

Fernando Bordignon y
Alejandro A. Iglesias

unipe: editorial universitaria

HERRAMIENTAS
SERIE TIC

Diseño y construcción de objetos interactivos digitales *Experimentos con la plataforma Arduino*

Fernando Bordignon y Alejandro A. Iglesias

unipe: editorial universitaria

HERRAMIENTAS
SERIE TIC

Bordignon, Fernando
Diseño y construcción de objetos interactivos digitales: experimentos con la plataforma Arduino / Fernando Bordignon; Alejandro Adrián Iglesias. -1a ed.-
Gonnet: UNIPE: Editorial Universitaria, 2015.
Libro digital, PDF - (Herramientas . TIC)

Archivo Digital: descarga y online
ISBN 978-987-3805-12-7

1. Educación Tecnológica. 2. Innovación Tecnológica. 3. Electrónica. I. Iglesias, Alejandro Adrián II. Título
CDD 005.1

UNIPE: UNIVERSIDAD PEDAGÓGICA
Adrián Cannellotto
Rector

Carlos G. A. Rodríguez
Vicerrector

UNIPE: EDITORIAL UNIVERSITARIA
Directora editorial
María Teresa D'Meza

Editor
Juan Manuel Bordón

Diagramación y diseño de maqueta
Verónica Targize

Camino Centenario nº 2565 - (B1897AVA) Gonnet
Provincia de Buenos Aires, Argentina
www.unipe.edu.ar

Esta edición se publicó en el mes de diciembre de 2015.

ISBN: 978-987-3805-12-7.

Agradecimientos

A las autoridades de la Universidad Pedagógica de la Provincia de Buenos Aires (UNIPE), por confiar en nuestro proyecto y darnos el apoyo necesario para concretarlo.

A todos los profesores y jóvenes que este año participaron del espacio de trabajo y reflexión que se constituyó en la ciudades de Lobos y Escobar. Gracias a sus enriquecedores aportes pudimos completar esta obra y también darle gran parte del sentido que tiene.

Finalmente y de forma especial queremos agradecer a los profesores Víctor Furci, Ángela Hahn y Oscar Trinidad, de la UNIPE, con quienes compartimos momentos muy buenos de aprendizaje y camaradería.

Fernando Raúl Alfredo Bordignon y Alejandro Adrián Iglesias

Índice

ANEXO. GUÍA DE PROGRAMACIÓN

Prólogo

El mundo actual está configurado, desarrollado y controlado, en gran parte, por el *software*. En este contexto, aquellos ciudadanos que no posean una serie de conocimientos, aptitudes y saberes prácticos que les permitan moverse con cierta libertad de pensamiento y acción posiblemente constituyan una nueva clase de "analfabetos" de época.

El poder interactuar con la sociedad actual es el centro del problema educativo y a la vez el elemento clave que permite dar libertad a los ciudadanos, para que se desarrollen de forma plena y se inserten de manera apropiada en el mundo laboral y social. Las políticas públicas de los últimos años han estado en función de desarrollar un entorno favorable que ha reducido la brecha de acceso a dispositivos, contenidos y servicios de red. Más allá de los innumerables problemas, contratiempos o desentendimientos, se han podido implementar acciones tendientes a brindar igualdad de oportunidades a los ciudadanos y a revalorizar la escuela pública, que aun con sus más y sus menos, se encuentra en pleno proceso de apropiación de los medios y las pantallas digitales.

Las tecnologías digitales abiertas relacionadas con el *hardware*, tales como las máquinas de impresión 3D y las placas de desarrollo electrónico tipo Arduino –de libre distribución y muy bajo costo–, han salido de las oficinas de investigación y desarrollo de las empresas y están penetrando en colectivos de personas comunes interesadas en el diseño y la creación de nuevos objetos digitales interactivos. ¿Por qué entonces no integrarlas a los procesos formativos de los jóvenes? Como indica el profesor Mitchel Resnick del Massachusetts Institute of Technology (MIT), el conocimiento en sí mismo ya no es suficiente debido a que en el mundo de hoy los cambios rápidos y continuos hacen necesario dar soluciones creativas a problemas inesperados. En sus palabras, "el éxito se basa no solamente en lo que usted sabe o en qué tanto sabe, sino más bien en su habilidad para pensar y actuar creativamente" (Resnick, 2009).

Con estas tecnologías de diseño y creación digital, los estudiantes pueden entender, a partir de experiencias de desarollo de objetos digitales interactivos, lo que ellos mismos son capaces de hacer con ideas propias y poderosas. La propuesta es un giro de actitud hacia la filosofía del hacer, pero en un nuevo ambiente de estudio donde la fabricación digital expande significativamente

sus posibilidades de expresión. La ampliación de oportunidades, a partir de un aprendizaje basado en experiencias significativas centradas en la construcción de objetos, está dando un nuevo impulso a las necesidades de conocer, en este caso, al conocer creando y haciendo. Esta experiencia ayuda a valorar cómo el aprendizaje también es una consecuencia del trabajo en la resolución de problemas concretos.

El profesor Gary Stager, investigador de las comunidades *maker*, propone incorporar estas actividades a la escuela, argumentando que "la cultura del hacer disuelve las distinciones entre ámbitos tales como las artes, las humanidades, la ingeniería y la ciencia. Más importante aún, rompe la escisión destructiva entre la formación profesional y la académica" (Stager, 2014b). Desde su punto de vista, las aulas donde se desarrollan tales actividades son espacios activos en los que se hallan estudiantes comprometidos, trabajando en varios proyectos a la vez; donde los maestros no sienten miedo de renunciar a un papel autoritario, dado que experimentan roles de mentor, estudiante, colega y experto. De algún modo, al trabajar de esta forma se cumple el rol docente que Seymour Papert (1996) indicara, con un maestro que crea las condiciones para la invención en lugar de dar información ya procesada.

Hoy estamos en condiciones de ir más allá de nuestras pantallas, dejando usos pasivos o guionados de la tecnología, para incluir en la educación formal saberes que ayuden a los estudiantes a desarrollarse y expresarse de forma plena en los tiempos que corren. Llegamos a un estadio de la sociedad en el cual pensar de forma creativa se ha convertido en un requisito indispensable para el desarrollo. Casi todo cambia tan rápido que los estudiantes tienen que encontrar cómo dar soluciones innovadoras a problemas nuevos o desconocidos. En este sentido, solo tendremos un futuro prometedor e interesante si somos capaces de dejar atrás la etapa de fascinación por las herramientas y los objetos tecnológicos. Si pasamos a un estadio superior, podremos utilizarlos para ampliar nuestra capacidad para expresarnos y construir conocimientos, inmersos en nuevas comunidades que innoven y compartan.

Este libro está en consonancia con los conceptos e ideas anteriores. Esperamos que sea de utilidad a profesores y estudiantes, que enriquezca sus prácticas de enseñanza y de aprendizaje en un ambiente creativo y abierto.

Primera parte
Arduino y el universo de los objetos digitales

CAPÍTULO 1

Hacia un mundo de objetos digitales interactivos

1.1. LOS SISTEMAS EMBEBIDOS EN LA VIDA COTIDIANA

La realidad que vivimos día a día se encuentra fuertemente ligada a la evolución de las tecnologías de la información y de la comunicación (TIC), así como a las políticas que se definen en torno a ellas. La economía global, la sociedad y la cultura se ven afectadas por el desarrollo de novedosas formas de comunicación que, a su vez, definen nuevas formas de acceso a la información, de relacionarse con las personas y de organización.

En gran medida, la red de redes es el escenario donde confluyen los servicios, los usuarios y la información que se produce y se ofrece. Actualmente, la interactividad con el mundo virtual se da en mayor medida a través de teléfonos inteligentes (smartphones) o computadoras personales. Los servicios de la red se alimentan de los datos que ingresan los usuarios, de sus propias costumbres y de la interacción que surge entre los distintos actores involucrados.

Sin embargo, desde hace un tiempo, este escenario está volviéndose aún más complejo. Miles de millones de microprocesadores se fabrican por año en el mundo, pero solo aproximadamente un 1% de ellos son creados para computadoras de propósito general como *notebooks*, *smartphones*, *tablets*, equipos de escritorio y servidores. El otro 99% de los microprocesadores permanece invisible, aunque paradójicamente –al mismo tiempo– a la vista de todos. Su destino son los sistemas embebidos: computadoras pequeñas y específicas que se encuentran en microondas, en sistemas automáticos de freno (tipo ABS) en automóviles, en semáforos de calle y en lavarropas; en tomógrafos para uso médico, en equipos de ultrasonido de aplicación múltiple, en equipos de audio, en ruteadores inalámbricos para acceso a Internet, en lectores de tarjetas de débito o crédito, en las terminales de cobro del transporte público y en muchos pero muchísimos objetos más con los que interactuamos a diario.

De hecho, la tendencia es que estos sistemas embebidos se vuelvan no solo más extendidos, en lo referente a variedad de objetos que los contienen, sino también más pequeños y más complejos, mientras que sus costos también disminuyen. No es un pronóstico "alocado" pensar que dentro de 5 o 10 años cada uno de los artefactos de nuestro hogar se encontrará conectado a Internet, en constante diálogo con otros artefactos, con el usuario, con los fabrican-

tes, con los proveedores y con quien lo quiera y pueda escuchar. Sin ir mucho más lejos, en el área de salud personal es probable que contemos, en nuestro cuerpo, con algún dispositivo añadido de forma permanente que nos permita monitorear nuestro estado de salud las veinticuatro horas del día, con la posibilidad de intervención remota automática.

Estadísticas presentadas por Cisco, una empresa proveedora de tecnología de *hardware* y *software* para comunicaciones, revelan que en el año 2003 había aproximadamente un dispositivo conectado a Internet por cada 13 personas en el planeta. En el año 2008 se produjo un crecimiento en esta relación, con algo más de 1,5 dispositivos por persona. Para el año 2015, el estimado es de 3,47 dispositivos por cada persona, y para el 2020 se estima que sean 6,58 (Evans, 2011). Con estas cifras queda clara la tendencia a una mayor densidad de dispositivos conectados a Internet en una sociedad donde los flujos de datos son mayores y donde la producción y almacenamiento de datos masivos también son parte de esta realidad.

El tamaño de la red Internet y la materia prima con la cual operan las tecnologías de la información y las comunicaciones crecen día a día generando un ecosistema de cambios en constante retroalimentación y aceleración. En este contexto los sistemas embebidos toman aún mayor importancia, siendo grandes protagonistas tecnológicos de lo que se espera que sea la próxima revolución.

En la Revolución Industrial, el objetivo principal de las tecnologías era aumentar las capacidades físicas del ser humano creando máquinas y procesos que potenciaran nuestra fuerza, velocidad y precisión en tareas mecánicas. La revolución de las TIC se mueve en un sentido mucho más profundo, ya que desarrolla tecnologías que potencian nuestras capacidades cognitivas. Los atributos por mejorar son ahora nuestras formas y posibilidades de percibir, recordar, aprender, organizarnos y comunicarnos. Las TIC, sin ninguna duda, están repercutiendo en cada uno de los aspectos de la vida humana.

Hoy en día, en una sociedad con un alto grado de desarrollo de las tecnologías digitales, donde casi todo está automatizado –en función de la supervisión, el control, la comodidad y la "eficiencia"–, los sistemas embebidos están en cada ámbito donde el ser humano vive, transita, se divierte y trabaja. Básicamente, existen para cooperar con nuestro confort y darnos una mayor calidad de vida. Es por eso que conocerlos, saber sus posibilidades y hasta animarse a experimentar con ellos es algo atrayente y a la vez enriquecedor a la hora de ampliar nuestro conocimiento y relación con el mundo.

1.2. ¿QUÉ SON LOS SISTEMAS EMBEBIDOS?

Un sistema embebido es una combinación de *hardware* y *software* que trabaja junto con algún sistema mecánico o electrónico diseñado para cumplir una función específica. Por lo tanto, se usa para dotar de "inteligencia" a un artefacto.

La diferencia principal entre un sistema embebido y una computadora de propósito general (por ejemplo, una *notebook*) radica justamente en la especialización que tiene el sistema embebido. Una computadora personal podría utilizarse para varias cosas: para jugar un videojuego, ver videos en Internet, comprar acciones, realizar una simulación o ejecutar algún programa de edición de imágenes. Por el contrario, un sistema embebido solo está diseñado, desde su *hardware* y *software*, para cumplir con un conjunto finito de funciones que determinan una tarea específica. Por ejemplo: calentar comida en un microondas, administrar eficientemente el sistema de frenos en un automóvil o ejecutar programas de lavado en un lavarropa.

En las radios u otros viejos equipos electrónicos, "la inteligencia" o la lógica de control del artefacto se localizaba en un circuito básico, compuesto de diodos, resistencias, capacitores y transistores. La complejidad de los aparatos, la velocidad con que se producen nuevos productos y el abaratamiento de costos de los microprocesadores hacen que actualmente sea más barato, y por sobre todo versátil, el desarrollo de sistemas embebidos dispuestos a dar soluciones a problemas o situaciones específicas. En un dispositivo cuyo control es analógico, realizar un cambio a veces significa tener que rediseñar desde cero el aparato (con los consiguientes costos y mayores tiempos asociados); en contrapartida, con un sistema embebido, actualizar la lógica de control es trivial (en términos de *hardware*), ya que solo se debe cambiar el programa que ejecuta el microprocesador.

El *hardware* de un sistema embebido se compone principalmente por un microprocesador, sensores y actuadores. Los sensores le permiten al microprocesador obtener información del mundo real; el microprocesador toma decisiones basado en el *software* que ejecuta; y los actuadores realizan acciones con el mundo físico.

Los sensores pueden ser variados. Hay de humo, de temperatura, de sonido, de proximidad, de velocidad, de posición, de tiempo, de inclinación y hasta sensores de ritmo cardíaco y presión atmosférica. Mientras que los actuadores pueden ser sirenas, motores, luces o controles de válvulas, entre otros.

El diseño y construcción de sistemas embebidos en ámbitos profesionales requieren de habilidades técnicas en electrónica y en programación muy desarrolladas. Lo que se busca en la producción a gran escala de estos sistemas es la eficiencia en costos y en consumo energético. Sin embargo, la complejidad y el poder de cómputo de los microprocesadores están permitiendo cada vez más el uso de lenguajes de alto nivel de abstracción, facilitando así el desarrollo de *software* para sistemas embebidos. Diseñar y crear prototipos de nuevos productos hoy es mucho más fácil. Esto permite construir en poco tiempo y con un bajo nivel de dificultad, dispositivos funcionales utilizando herramientas de más fácil acceso, tanto desde la programación como desde la creación de *hardware*. En la actualidad, esto se vehiculiza gracias a que existen diversas plataformas de diseño y prototipado rápido que están al alcance no solo de profesionales sino también de aficionados y, más importante aún, de cualquier ciudadano.

1.3. LOS SISTEMAS EMBEBIDOS SE ESCAPAN DE LA INDUSTRIA Y LLEGAN AL HOGAR

La necesidad de crear sistemas embebidos de forma rápida y sencilla impulsó el desarrollo de diversos microprocesadores y placas que buscaron diluir las limitaciones técnicas anteriores que representaba crear nuevos sistemas. Este camino, en consecuencia, condujo a la creación de plataformas de fácil acceso no solo para profesionales y estudiantes del sector, sino también para un público más amplio. Si bien al día de hoy esta línea de *hardware* tiene más visibilidad gracias a sistemas como Arduino,[1] también posee un extenso desarrollo previo.

En 1975 (cuatro años después del desarrollo del primer microprocesador en un chip, el Intel 4004) se creó el primer microprocesador PICaxe. A diferencia del popular chip de Intel, el PICaxe apuntaba a un público que no era estrictamente profesional. Estaba orientado a posibilitar que estudiantes y *hobbistas* puedan usarlo en sus propios experimentos y creaciones electrónicas. Sin embargo, su punto débil era que utilizaba un lenguaje de programación poco accesible y bastante complejo para principiantes, orientado a un sector específico que tenía interés en dedicarse al desarrollo de sistemas embebidos de manera profesional en el mediano o largo plazo.

Luego, a principios de la década de 1990, se presentó una nueva plataforma con similares objetivos. Stamp también estaba diseñada para estudiantes y *hobbistas*, pero por primera vez se utilizaba un lenguaje de programación de alto nivel, haciendo mucho más accesible la experiencia de programarla. Sin embargo, su costo era algo elevado, tenía un microprocesador bastante limitado y el entorno de desarrollo con el cual se podía programar la placa solamente podía ser ejecutado bajo el sistema operativo Microsoft Windows.

En el año 2005, como respuesta a estos problemas, un grupo de estudiantes del Instituto Ivrea –en la ciudad italiana del mismo nombre– diseñó y creó la placa Arduino. El objetivo de esta placa era el desarrollo de una plataforma que facilitara la creación de artefactos interactivos digitales por parte de estudiantes, *hobbistas*, artistas y público en general, basándose por primera vez en conceptos derivados del *software* y el *hardware* libre. Las interfaces de programación y el *software*, a diferencia de sus predecesores, resultaron muy fáciles de aprender y de utilizar. Por otro lado, su construcción se liberó a la gente, dado que cualquiera era libre de descargar los diseños de la placa Arduino y de construirla en su propio hogar. Dentro de los objetivos de diseño de la placa, siempre estuvo presente el bajo costo, lo cual en la actualidad la ha hecho muy accesible para instituciones educativas y particulares.

Puesto en contexto, Arduino, con su diseño de fácil acceso, sus interfaces de programación sencillas y la filosofía de *software* y *hardware* libre, se relaciona perfectamente con los colectivos *maker* (de "los que hacen"), espacios Fab Lab (laboratorios de fabricación digital) y con comunidades

1. Sitio proyecto Arduino: https://www.arduino.cc/

en línea DIY (*Do It Yourself*), que promueven las actividades de tipo hágalo usted mismo. Por eso, gran parte de su popularidad y aceptación se dio y todavía se apoya en tales comunidades, donde aprecian las facilidades de soporte que tiene la placa.

En concreto, es gracias a estas comunidades que plataformas como Arduino y otras semejantes cuentan con muchos recursos de demostración, soporte y apoyo creados por los propios usuarios. Estos ayudan a los principiantes, y a los que no lo son tanto, a desarrollar sus proyectos sumergiéndose en el mundo de la creación de artefactos interactivos digitales, guiados por sus propios intereses y su propio ritmo. En los foros, grupos de redes sociales, blogs y repositorios de materiales compartidos –creados, en gran parte, colaborativamente– es posible encontrar las instrucciones para desarrollar desde estaciones climáticas, automatizaciones del hogar y marionetas electrónicas hasta sistemas de seguridad, de riego automático e inagotables ejemplos más.

1.4. LOS SISTEMAS EMBEBIDOS Y LA "INTERNET DE LAS COSAS"

Los sistemas embebidos, además de dar la posibilidad de dotar de inteligencia a una gran diversidad de aparatos, están adquiriendo aún más importancia gracias a un concepto conocido como "Internet de las cosas" (IoT por su sigla en inglés, *Internet of Things*).

La reducción de costos de fabricación de diversas tecnologías digitales y al desarrollo de nuevas plataformas de *hardware* permite prever que en un par de años el costo de conectar cualquier objeto a Internet será inferior a un euro (Tirado Fernández, 2015). Esto abre una gran oportunidad para que muchos objetos de uso cotidiano puedan estar dotados de una conexión a Internet para interactuar con el mundo que los rodea y así comunicarse con otros objetos, con diferentes servicios y con sus propietarios y usuarios. En este sentido la IoT representará una gran revolución para el desarrollo de la red ya que los humanos dejarán de ser intermediarios en los canales desde los cuales la red de redes toma sus datos, y permitirá que algunos dispositivos tomen sus decisiones de manera más autónoma.

Los usos de los dispositivos IoT no se limitan a la electrónica de uso diario como lavarropas, cocinas, heladeras y diversos entretenimientos hogareños, sino que abarcan también un gran espectro de aplicaciones que incluyen su empleo en automóviles y autopartes, en diversos vehículos y medios de transportes, para autochequeo y control de stock de productos en góndolas de mercados, en maquinarias industriales de producción e incluso en agricultura y cría de animales. Un ejemplo de experiencias recientes es su uso con ganado: se calcula que una vaca monitoreada por un sistema embebido produce un total aproximado de 200 megabytes de información al año, que incluyen datos sobre su estado de salud, de alimentación y su localización (Evans, 2011).

En consecuencia, la cantidad de información nueva que se genera día a día en Internet crecerá aún más y las bases de datos se verán, más que nunca,

atiborradas de información. No solo crecerá la cantidad de datos sino que además estos serán más diversos (variedad de sensores y dispositivos), más exhaustivos (los sistemas embebidos no olvidarán completar datos), exactos (no se equivocarán) y actualizados (estarán conectados a Internet comunicándose todo el tiempo).

Son muchas las organizaciones y empresas que ya están usando esta gran cantidad de datos para crear predicciones, reconocer patrones de consumo y de comportamiento, y muchas aplicaciones más. Las tecnologías que se utilizan para realizar estas tareas con grandes volúmenes de información son las que se conocen como datos masivos o, en inglés, *Big Data* (Bollier, 2010). Para citar ejemplos de usos actuales de este tipo de herramientas se tiene el caso de la empresa Google, la cual a través del análisis de los archivos de registros (*logs*) de las consultas hechas en su motor de búsqueda y en base a datos de geolocalización de dichas consultas, ha logrado predecir posibles brotes de gripe y alzas en los índices de desempleo antes de que las estadísticas oficiales se hagan públicas. Por su parte, las empresas de tarjetas de crédito utilizan los datos de las compras de los clientes –y también de geolocalización– para reconocer patrones de consumo. De ese modo alimentan estrategias de marketing y también refuerzan sus normas de seguridad, ya que logran bloquear con cierta anticipación las tarjetas de crédito y así evitan fraudes antes de que siquiera pueda efectivizarse una compra.

Las herramientas que provee el procesamiento de datos masivos están diseñadas para descubrir patrones que están ocultos en los datos utilizando técnicas de correlación estadística y diversos algoritmos de inteligencia artificial. Gracias a los datos que se podrán generar con IoT, se espera que se puedan descubrir patrones y relaciones que contribuyan a un uso más eficiente de los recursos energéticos, de logística de alimentos y medicinas, del tráfico, para la prevención de desastres naturales y control de enfermedades y plagas. Se está creando una sinergia entre *Big Data* e IoT, dado que se alimentan mutuamente. Sin embargo, los problemas que plantean estas tecnologías también se ven potenciados, ya que no están aún del todo claras cuestiones como la privacidad de los datos y las implicaciones legales que pueda tener el uso, recopilación, análisis o transferencia de esa información.

Sin embargo, IoT tiene todavía varios obstáculos por superar para poder desarrollarse plenamente. Se considera que estamos recién al inicio de esta nueva era. Por un lado, es necesaria la creación de estándares que regulen las comunicaciones y la infraestructura sobre las cuales funcionan los dispositivos. En este punto, organizaciones internacionales sobre el desarrollo de estándares y recomendaciones de las telecomunicaciones, como la IEEE (Institute of Electrical and Electronics Engineers) y la ITU (International Telecommunication Union), y grandes empresas importantes del sector de las comunicaciones, como Cisco e Intel, han estado trabajando activamente en la definición de dichas normas desde hace un tiempo.

Otro punto importante es avanzar en la implementación de la reciente versión del protocolo de red de Internet, puesto que la reciente versión IPv6 per-

mitirá identificar unívocamente los dispositivos en la red, sin la necesidad de equipos intermedios. En la actualidad el protocolo predominante es la versión IPv4, con un esquema de direccionamiento reducido y ya agotado. Por último, también es necesario el desarrollo de nuevas fuentes de energía autogestionadas para poder alimentar la gran cantidad de sensores y dispositivos que poblarán nuestras vidas.

El ecosistema en el que funcionan los sistemas embebidos es grande y complejo. Es responsabilidad del ciudadano de este nuevo mundo estar alerta y ser consciente de los avances que influyen en el desarrollo de nuestra sociedad. El uso de plataformas como Arduino y similares permite, además, una participación activa en el desarrollo de las nuevas tecnologías.

CAPÍTULO 2

Sistemas embebidos en la escuela

2.1. DISEÑAR Y CREAR EN EL MUNDO DIGITAL

El mundo actual se encuentra al final del tránsito hacia un nuevo modelo, que viene de la sociedad industrial y se dirige hacia la sociedad de la información. Esta última se basa en una nueva economía donde la generación de conocimientos toma un alto valor y las tareas repetitivas son desempeñadas, casi con exclusividad, por máquinas que a su vez son dirigidas por sistemas informáticos.

Hoy en día, no solo las tareas físicas en la producción de bienes se encuentran automatizadas: el avance de las TIC ha permitido una penetración mayor en muchos aspectos que incluyen tanto ámbitos productivos industriales como logísticos, organizacionales y de comunicación, afectando la forma de trabajo de todos los empleos. La competitividad en el terreno laboral se encuentra fuertemente ligada al espectro de conocimientos propios de la disciplina y, también, al desarrollo de saberes y habilidades en el manejo de las tecnologías informáticas que los potencian.

Por ejemplo, hoy un arquitecto necesita dominar las herramientas de diseño asistido por computadora (CAD) para poder realizar planos y proyectos que incluyan simulaciones visuales y estructurales. Un odontólogo debe saber dominar las herramientas de escaneo visual y modelización para crear esquemas y moldes de piezas dentales y planificar cirugías. También un distribuidor de alimentos necesita manejar tecnologías GPS para navegar por las calles, y usar servicios como Waze[2] para planificar las rutas teniendo en cuenta el estado de tránsito en tiempo real.

Las herramientas digitales facilitan y potencian el desempeño de las tareas de una profesión o de los quehaceres del día a día. Sin embargo, el valor agregado que una persona puede darle a esas tareas aún está dado por sus propios atributos. Las tecnologías diseñadas para mejorar capacidades cognitivas como recordar, registrar, representar, analizar y compartir información, se pueden entender como un vehículo que pondera las aptitudes de las personas. En este contexto la capacidad de síntesis, las actitudes emprendedoras,

2. Sitio Waze: https://www.waze.com

lúdicas, curiosas y responsables, además del potencial creativo, se ponen en función de crear soluciones a una amplia gama de problemas.

La creatividad es sin duda una de las capacidades que funcionan como catalizadoras en el momento de obtener, combinar y producir soluciones. Antiguamente se la pensaba como un elemento innato de las personas; sin embargo, Howard Gardner (2000) indica que debe pensarse como surgida "de la interacción de tres nodos: el individuo con su propio perfil de capacidades y valores; los ámbitos para estudiar y dominar algo que existen en una cultura; y los juicios emitidos por el campo que se considera como competente dentro de una cultura. En la medida en que el campo acepte las innovaciones, una persona o su obra puede ser considerada creativa; pero si las innovaciones se rechazan, malinterpretan o juzgan poco novedosas, resulta inútil seguir sosteniendo que un producto sea creativo". La creatividad no solo está asociada a la inteligencia pues conlleva elementos que configuran la personalidad del individuo, así como otros del ámbito y del campo presentes en la sociedad en general, y además requiere de un ecosistema para fomentarla y permitir su maduración.

Con referencia a la escuela y la creatividad, Gardner asevera que la mente de un niño de cinco años representa la cumbre del poder creativo, dado que a los niños les gusta explorar, son entusiastas ante las cosas nuevas y tienen una gran capacidad imaginativa. Pero la capacidad de conservar este espíritu creativo depende en gran parte de los mensajes que reciben desde afuera (Gardner, 2008). Es importante por lo tanto que el sistema educativo acompañe y fomente el desarrollo de la creatividad y las actitudes lúdicas. Sin embargo, las instituciones más antiguas y menos afectadas por los movimientos y ajetreos del mundo del mercado son aquellas a las que más les cuesta adaptarse a los cambios en los modelos de sociedad. En general, al no verse amenazada en el corto plazo por los cambios sociales y tecnológicos, la escuela como institución no ha tenido la misma presión que muchas de las empresas que se disputan codo a codo el terreno económico.

Entre las posiciones más críticas al sistema educativo contemporáneo se destaca la correspondiente a Dale Stephens (2013), quien argumenta que "los sistemas y las instituciones que vemos a nuestro alrededor, en las escuelas, las universidades y el trabajo, están siendo sistemáticamente desmantelados. Si alguien desea aprender las habilidades necesarias para navegar por el mundo, con todo su ajetreo, su conectividad y su creatividad, tendrá que *hackearse* su propia educación". En este sentido, Stephens plantea que un camino posible para que el ciudadano complete su educación es apoyar las vías formales a través de aprendizajes en medios no tradicionales. De alguna manera, en ese texto el autor deja ver las tensiones actuales que derivan en críticas centrales a un sistema educativo que –según él– no prepara a los estudiantes para afrontar los nuevos escenarios que están configurando a nuestra sociedad. En esta línea, hace más de una década, Idit Harel (2002) se refería a las habilidades del futuro como las relacionadas con el dominio de las "tres equis" –eXploración, eXpresión e intercambio, *eXchange* en inglés–

y a que los medios digitales deben permitir que los niños incorporen otras dimensiones en relación al aprendizaje creativo, expresivo e imaginativo.

Desde otra perspectiva, el experto en tecnología Jaron Lanier (2011) tiene una mirada escéptica con respecto a la creatividad, argumentando que, si bien el espacio web en sus inicios potenciaba la creatividad individual, a partir de la masificación del uso de las redes sociales esto no pasa, dado que se fomenta la mentalidad de rebaño. Si bien Lanier plantea una postura provocativa, la realidad probablemente se encuentre en alguno de los matices de este complejo panorama, dado que los colectivos de aprendizaje han demostrado ser espacios enriquecedores de intercambio y desarrollo. Además, independientemente del grado de acierto que tenga, Lanier no deja de recalcar la necesidad de contar con un acompañamiento para saber aprovechar y explotar las posibilidades que plantean las nuevas tecnologías.

Este estado en el que vivimos, caracterizado por la abundancia y la riqueza de elementos tecnológicos digitales, da la oportunidad de desarrollar nuestro pensamiento creativo de forma intensa; las posibilidades de innovar y crear son muchas. No estamos solos frente a esta tarea, hay colectivos importantes que nos apoyan y a la vez nos dan insumos para mejorar nuestras posibilidades día a día. Gradualmente, a lo largo y ancho del planeta, las tecnologías de la información y la comunicación están despertando un nuevo panorama, donde el *emprendedorismo* de la gente y la consiguiente creación de servicios y productos se configuran más como una realidad que como una posibilidad. Pero para dar un mayor impulso a esta situación se necesitan, desde las escuelas, nuevos enfoques en relación a los procesos de enseñanza y de aprendizaje, puesto que el objetivo principal debería ser lograr una sociedad de ciudadanos creativos que diseñen y construyan nuevos elementos en función de sus propias necesidades y de aquellas de las comunidades a que pertenecen.

La clave es poder cambiar una mentalidad impuesta desde la Revolución Industrial, aquella que marcó las categorías bien diferenciadas de productores y consumidores. Ahora se deben repensar las reglas impuestas por la cultura, para reordenarlas y resignificarlas en función de crear objetos y servicios ajustados a nuestras necesidades. El éxito de esta forma de pensar y de hacer seguramente estará coronado por la extensión del conocimiento y una relación mucho más genuina con el mundo en que nos toca vivir.

2.2. LA TECNOLOGÍA COMO FORMA DE EXPRESIÓN

Dado que no toda experiencia con pantallas digitales es absolutamente pasiva, la interactividad es definida como una suerte de diálogo en el que los usuarios se expresan con las aplicaciones y los servicios. De esta manera, la interactividad es vista como la capacidad de participación del usuario en un proceso comunicativo. Esto implica que las personas que interactúan con los medios digitales dialogan con los textos, las imágenes y, por ende, con sus autores, hasta el nivel de desarrollo de poder convertirse ellos mismos en creadores de contenidos multimodales.

Las interfaces de los usuarios con las computadoras han ido creciendo en prestaciones y en adaptación a las necesidades de prestación: en los inicios, se tenían tarjetas perforadas para ingresar y almacenar datos e instrucciones; después, aparecieron los teclados y monitores; en la actualidad, la experiencia de interacción pasa por dispositivos múltiples que se empiezan a adaptar a nuestros sentidos, pensamientos y movimientos corporales.

En los últimos años, también el concepto de interactividad se ha ampliado, dado que en el origen del espacio web el usuario solo disponía de opciones asociadas a la navegación sobre un sitio. Ahora dispone de una amplia gama de herramientas y servicios en línea que complementan y amplían su participación en la web, promoviendo sus posibilidades creativas, productivas y participativas, ya sea desde una posición individual o colectiva.

El punto de partida de este empoderamiento del usuario, a partir de la ampliación de sus capacidades, se dio con la llegada de la web 2.0. Tal como indica David Casacuberta (2003), profesor de filosofía de la ciencia en la Universidad Autónoma de Barcelona, "la verdadera interactividad es la creatividad. Escoger el final de una película no es interactivo. Desarrollar todo el guión a nuestro gusto, sí". De este modo, la creatividad –potenciada por los nuevos medios– es el espacio de desarrollo de las personas en pos de una ampliación de sus posibilidades de expresión y de interacción con el mundo. Ha sido tal la importancia de este tema que Manuel Castells (2001) abogó por el establecimiento de una pedagogía "basada en la interactividad, personalización y el desarrollo de la capacidad de aprender y pensar de manera autónoma".

El nivel de interacción del usuario promedio con las tecnologías digitales se ha ido complejizando con el tiempo y, en sentido inverso, los conocimientos necesarios para producir nuevos contenidos se han hecho cada vez más simples. En este contexto las posibilidades de creación por parte de los usuarios también crecieron, pero siempre contenidas en la infraestructura de *hardware* y encerradas detrás de las pantallas y en el plano de la virtualidad. Las posibilidades de materializar esas creaciones digitales estaban hasta ahora reservadas a sectores industriales o laboratorios empresarios.

El desarrollo de tecnologías como Arduino y otras placas similares ha hecho posible escapar a este encierro virtual creando el ecosistema necesario para generar proyectos que puedan interactuar con el mundo físico. Este puente entre el mundo virtual y el físico permite la creación de una gran variedad de elementos y tecnologías, abriendo nuevos caminos y dando la oportunidad de que los usuarios no solo generen contenidos para Internet y sus diversos servicios y redes, sino también que diseñen soluciones y productos a la medida de sus intereses y necesidades particulares. Estas tecnologías permiten dotar de "inteligencia" a diversos proyectos de diseño de objetos digitales, capturar y guardar datos del mundo físico, procesarlos y realizar acciones a partir de las decisiones producidas por una lógica digital automática creada por el ciudadano. Las posibilidades que se abren con proyectos de estas características abarcan a su vez un amplio abanico de prototipos que van desde sistemas de riego automático para una huerta hogareña, a sistemas de seguridad caseros,

instrumentos musicales electrónicos, estaciones meteorológicas, automatizaciones en el hogar, proyectos de robótica, interfaces alternativas para computadoras, sistemas de registros varios mediante uso de tarjetas RFID (almacenamiento y lectura de datos por radiofrecuencia) como las tarjetas por contacto del transporte público, y muchísimos ejemplos más.

El desarrollo de esta clase de proyectos, los cuales poseen un alta interacción con el mundo físico, tiene asociada además la multidisciplinariedad. En ellos se combinan y entran en juego variedad de saberes y conocimientos que se relacionan con lo técnico, lo físico, lo mecánico y las prácticas ingenieriles, con problemas prácticos, de factibilidad y, en definitiva, con lo propio de cada problema en particular. En este aspecto los proyectos requieren de una mayor interrelación entre estas habilidades y conocimientos, dado que presentan muchas y diversas dificultades a resolver. A su vez, los participantes desarrollan habilidades vinculadas con la investigación en búsqueda de soluciones o en la práctica de análisis de factibilidad, dotándolos de nuevas herramientas intelectuales necesarias para resolver dichos problemas, las cuales también son extrapolables a diversas actividades de su vida cotidiana.

En esta misma línea, la fabricación digital, entendida como la posibilidad de materializar objetos diseñados digitalmente, acorta aún más la brecha entre el mundo virtual y el físico, así como el mundo de la producción y el del consumidor. Diseñar objetos digitalmente, poder compartirlos y modificarlos a través de Internet, hace más importante el papel del ciudadano en la creación de nuevos productos y tecnologías puesto que lo lleva a niveles de participación nunca antes establecidos.

Más importante aún que los proyectos en sí es la posibilidad de compartir no solo el producto final en una red social o en una comunidad de creadores, sino también el conjunto de sus procesos, motivaciones y dudas. Esto genera una sinergia con la comunidad en Internet, donde un proyecto puede ser la motivación inicial para otro distinto.

La personalización de objetos imprimibles y su diseño, la posibilidad de dotarlos de inteligencia con sistemas como Arduino, y las comunidades en línea que permiten compartir y remezclar los proyectos, crean un ecosistema complejo que está en constante evolución y retroalimentación. El protagonismo de todo este movimiento, si bien está rodeado y empoderado por tecnologías, se centra no en las técnicas ni en los objetos de *hardware* o *software* que las rodean, sino por el contrario en los aspectos humanos más esenciales: el aprendizaje en comunidad y el desarrollo de soluciones múltiples a problemas complejos.

Este es el mundo en el que el ciudadano tiene que abrirse camino hoy y del cual debe ser capaz de sacar el máximo provecho. Este es a su vez el ciudadano que el sistema educativo actual debe mirar con especial atención.

2.3. LA ESCUELA AL ENCUENTRO DEL MUNDO DIGITAL

Hoy el conocimiento aplicado se da a partir de la evolución de la fuerza humana, y se centra en las máquinas digitales automáticas que construimos. En este sentido se le atribuye un alto valor en la sociedad contemporánea y por ende también es un elemento central de discusión del poder, dado que la tensión actual se transforma a partir de los cambios en los modos de circulación del saber. Lo que alguna vez fue centralizado en pocas instituciones, fuertemente controlado y protegido, con acceso a muy pocas personas formadas, hoy ya no lo es.

Los modos de circulación del saber empezaron a cambiar radicalmente en las últimas décadas, desde la aparición de los medios masivos analógicos de comunicación, y ha tenido una ruptura significativa (con las viejas formas) a partir de la expansión de las redes digitales de información. El filósofo Jesús Martín-Barbero (2003) ha reflexionado de manera profunda sobre los cambios producidos por los medios y las redes globales en lo referente a cómo las personas acceden a la información y construyen conocimientos, indicando que las instituciones más afectadas por tales cambios son la familia y la escuela. En este sentido su pensamiento es coincidente con el de McLuhan (1968), quien anticipadamente percibió la llegada de un aula sin muros como consecuencia de los cambios que se estaban dando. Esta ruptura en los modos de circulación del saber se ha producido a través de una serie de hechos que Martín-Barbero denominó descentramiento, deslocalización, destemporalización y diseminación.

El descentramiento se produce cuando las fuentes de información empiezan a circular por fuera de los lugares clásicos donde estuvieron "custodiadas" durante siglos. El saber deja de tener por única residencia al libro gutemberiano; por primera vez desde la aparición de la imprenta se produce una ruptura fuerte en la forma en la que circulan y en dónde residen los saberes. Ya la escuela y la biblioteca no son los únicos espacios de referencia, los saberes fluyen por otros caminos exteriores que enriquecen de forma significativa las posibilidades de acceso instantáneo, independientemente de la ubicación física de aquellos que proveen y demandan el servicio de consulta. Para que este fenómeno de descentramiento suceda, más allá del establecimiento de redes de datos, antes fue necesario que se pusieran a punto técnicas y aparatos de digitalización. Una copia digitalizada de un libro pasó a ser un objeto computable que pudo duplicarse, transmitirse, almacenarse y procesarse fácilmente para beneficio de su acceso y lectura por parte de ciudadanos comunes. Hoy queda claro que el libro no se reemplaza con la incorporación de nuevos contenidos y formas de acceso, distribución y lectura, propias de los medios digitales. Estas por el contrario amplían su función social y cultural al permitir un acceso potencial mucho más amplio por parte de ciudadanos comunes de casi cualquier país del mundo. En definitiva, lo que ha entrado en crisis es el modelo tradicional de trabajo escolar, que aseguraba en buena parte una reproducción fiel de la información y las habilidades más que el desarrollo de saberes.

Los conceptos de deslocalización y destemporalización están en relación con que ahora los saberes están por fuera de los espacios físicos y de los tiem-

pos tradicionales asociados con la distribución y el aprendizaje del saber. Las redes digitales globales han posibilitado nuevas formas de mediación por cuanto expanden y mejoran las oportunidades y procesos de enseñanza. La educación a distancia está demostrando su valor, dando nuevas oportunidades de formación básica y continua a ciudadanos a los que por razones de residencia o de edad se les hace difícil insertarse en el modelo educativo presencial. Más allá de que esta modalidad ayuda a mejorar situaciones de derechos postergados de mucha gente, también apoya los nuevos requerimientos de formación continua de las personas. Hoy está claro que aprender a aprender es una aptitud básica requerida y demandada por la nueva configuración de sociedad. Otra vez aparece la idea de que la educación se redefine de forma gradual, no se reemplaza radicalmente sino que se adapta lentamente a nuevos escenarios a través de la expansión de sus formas y métodos.

La diseminación del saber se da en función de los procesos de descentramiento y deslocalización, dado que tiene que ver con la caída de las barreras que separaban los conocimientos académicos del saber común. Para Martín-Barbero (2003),"la diseminación nombra el movimiento de difuminación tanto de las fronteras entre las disciplinas del saber académico como entre ese saber y los otros, que ni proceden de la academia ni se imparten en ella de manera exclusiva". El autor advierte a la vez sobre los riesgos de tener un mundo donde existan solamente conocimientos especializados, dado que a mayor cantidad de estos, mayores los riesgos para el hombre; pero aporta que una solución se puede dar "en la articulación de conocimientos especializados con aquellos otros que provienen de la experiencia social y de las memorias colectivas".

Los aportes anteriores sobre los cambios en las formas de circulación y acceso al saber dan cuenta de una realidad irreversible, impuesta de forma arrasadora por la expansión de los medios digitales en todo el planeta. Martín-Barbero ha aportado insumos valiosos de pensamiento que permiten establecer un marco para reflexionar sobre cómo seguir adelante en la redefinición de los conceptos de enseñanza y de aprendizaje.

El educador José Joaquín Brünner (2006) pone en una misma línea de diálogo las visiones tempranas de McLuhan junto a las apreciaciones posteriores de Martín-Barbero, al indicar que "el contexto en que opera la escuela, como los propios fines de la educación, están siendo transformados drástica y rápidamente por fuerzas materiales e intelectuales que se hallan fuera del control de la comunidad educacional y cuyos efectos sobre esta serán inevitables. [...] El conocimiento deja de ser lento, escaso y estable, y, por el contrario, está en permanente proceso de expansión y renovación. [...] También la especialización es cada vez más pronunciada y pulveriza el conocimiento hasta el infinito."

Es difícil pensar que el sistema educativo, destinado a promover y proveer mecanismos de enseñanza y aprendizaje, quede exento de experimentar transformaciones, incluso en el corto plazo. Más aún, en lo inmediato el principal protagonista de cambios en sus prácticas y concepciones serán el propio docente y, por ende, también los estudiantes.

Los sistemas informales de aprendizaje por los que atraviesan alumnos y profesionales se desenvuelven en espacios donde Internet, la colectividad, el interés propio, el aprender haciendo, el material actualizado y la búsqueda de reconocimiento de pares son la moneda corriente. Los textos que utilizan están "vivos" y sus autores responden al diálogo, muchas veces no son expertos en el tema, sino pares más experimentados, que cooperativa y colectivamente crean los contenidos que utilizan para desarrollar su aprendizaje y los comparten. Se utilizan variedades de recursos, y muchos materiales son el resultado de combinaciones de autores reconocidos formalmente o expertos tácitos con experiencia. Este aprendizaje es autoguiado e incentivado por intereses propios. La cultura del hacer y el experimentar es la que valida y fortalece estos aprendizajes. Las soluciones que se dan a los problemas tienen un fuerte contenido creativo ya que no responden a caminos preestablecidos.

Desde hace tiempo, las corrientes pedagógicas de Piaget y Vigotsky plantean que el aprendizaje no puede ser transmitido como si fuese simplemente un mensaje sino que, por el contrario, el conocimiento puede surgir solo como una construcción personal a partir de experiencias que se enriquecen y potencian con relaciones interpersonales. Esto da la pauta de que las tecnologías de las comunicaciones no son las responsables de crear este estilo de aprendizaje, sino que simplemente responden y potencian un proceso más intuitivo y natural de aprendizaje que se venía proponiendo desde mucho antes.

Sin embargo, la variedad de fuentes y autores, así como la complejidad misma de la red y los medios que en ella funcionan, hacen necesario el desarrollo de habilidades y competencias nada triviales: contar con un buen nivel de lectocomprensión y escritura, tener la capacidad de validar fuentes, autores, medios e información, tener autodisciplina en lo que se emprende, saber leer críticamente la información y poder identificar subjetividades y sesgos, poseer un marco de referencia fuerte sobre el cual construir conocimientos más complejos, saber distinguir lo importante en textos, poder sintetizar y producir contenidos, y por sobre todo saber mantener la motivación frente a dificultades, así como acostumbrarse a realizar y recibir críticas de forma adecuada.

Frente a estas necesidades, el sistema educativo formal no está inmovilizado sino que se halla en tensión enfrentando un nuevo desafío. No solo es necesaria la modificación y la actualización de las prácticas de enseñanza. También es importante incorporar algunos elementos de los sistemas de aprendizaje informales, los cuales están demostrando ser apropiados para el desarrollo de saberes en los jóvenes de este tiempo.

2.4. ENSEÑAR Y APRENDER HACIENDO: EL MOVIMIENTO *MAKER*

Linus Torvalds asegura que todas las motivaciones del hombre se pueden agrupar en tres categorías: "supervivencia", "vida social" y "entretenimiento"(Himanen, 2002). En este sentido, indica que la evolución está dada por el pasaje gradual entre cada una de ellas y que es eso lo que determina el verdadero progreso.

Para Torvalds un *hacker* es una persona que ha pasado y superado la etapa de usar su computadora para sobrevivir y está en los estadios siguientes de su esquema de progreso, en los que la persona usa la computadora para relacionarse con sus pares y construir comunidad y también le da usos relacionados con el entretenimiento.

Desde siempre han existido personas aficionadas a "hacer" cosas, ya sea creando, modificando o reparando objetos de su interés. Por ejemplo, en la jerga habitual de los españoles suele utilizarse el término "manitas" (*tinkerer* en inglés) para referirse a aquellas personas que se animan a arreglar equipos o artefactos sin tener una formación específica. Las caracteriza su capacidad de aprender, ver cómo funcionan ciertos objetos y solucionar problemas. En general, son personas con alta destreza manual. Otro término coloquial relacionado es "cacharreo", el cual denota una acción que tiene que ver con la reparación de algo sin que la persona que ejerce la acción sea necesariamente un técnico o profesional.

Hoy las culturas del bricolaje, del hágalo usted mismo, del *hacker* y del inventor se funden y se renuevan con las tecnologías digitales. Así se generan colectivos propios de este tiempo, donde la tecnología digital tiene una alta presencia. Los garajes de otro tiempo, espacios privilegiados donde el inventor, manitas o artesano se recluía en sus proyectos, hoy dan paso a colectivos de personas que se vinculan y cooperan tanto en nuevos espacios físicos como en comunidades virtuales de aprendizaje.

El colectivo *maker*, en principio, puede verse como una extensión del movimiento hágalo usted mismo a principios del siglo XXI, inspirado en la amplia circulación en los distintos medios de prácticas de fabricación, herramientas y grupos de soporte. De alguna manera, los métodos y herramientas que tradicionalmente estaban restringidos a laboratorios de investigación y al desarrollo de empresas salen de ellos y son incorporados por la gente común, que los empieza a recrear en clubes o en los garajes de sus casas para su propio beneficio. Desde su concepción el movimiento toma como insumo la energía creativa, la necesidad permanente de conocer y la imaginación de sus participantes.

Los colectivos de *makers*, cuando evolucionan y se consolidan en una ciudad o zona, toman forma en espacios físicos. A estos se los consideran lugares donde la gente puede crear, construir o fabricar a partir de sus ideas individuales o grupales. La dinámica particular que se da en tales espacios fomenta el juego y la exploración, propiciando prácticas enriquecedoras relacionadas con el aprender, el crear y el compartir.

La introducción y adopción de nuevas tecnologías digitales, tales como la impresión 3D y la minicomputadora Arduino, estimularon el movimiento *maker*. También lo fortalecieron las nuevas oportunidades brindadas por el prototipado rápido y las herramientas de fabricación digitales de bajo costo; un sistema de provisión de componentes y distribución directa de productos físicos en línea; y la participación creciente de todo tipo de personas en comunidades en red, reunidas por sus propios intereses y generalmente basadas en compartir metas comunes (Dougherty, 2013).

Hoy, desde la educación formal, se está prestando atención los *makerspaces* (espacios de reunión de los *makers*) dado que se piensa que pueden apoyar a los procesos de enseñanza y de aprendizaje en ciencia, tecnología, ingeniería y matemática a partir de sus actividades relacionadas con el desarrollo de habilidades prácticas.

Según Stager (2014a), gran parte de la población "ha comenzado a reconocer que el conocimiento es una consecuencia de experimentar y que la tecnología puede desempeñar un papel en la construcción del conocimiento. Esta revelación es un acto de construccionismo en sí mismo". Lo cual lo lleva a afirmar que "Papert no solo es el 'padre' del construccionismo, sino también del movimiento *maker*", del cual se deduce que potencialmente existe un puente, de carácter virtuoso, entre los colectivos de creadores y las necesidades de explorar las prácticas educativas en función de repensarlas y adaptarlas a los tiempos que corren.

Los aportes educativos de Papert se pueden sintetizar en ocho ideas principales, las cuales definen el espíritu del laboratorio construccionista (Stager, 2014a):

1) Aprender haciendo: en general aprendemos mejor cuando nuestras actividades nos parecen interesantes. La motivación y el interés son elementos esenciales del proceso educativo y si lo que construimos es lo que queremos, los resultados de aprendizaje serán mejores.
2) La tecnología como material de construcción: la tecnología posibilita construir múltiples cosas interesantes. En particular la tecnología digital, que ofrece un gran repertorio de herramientas.
3) Diversión difícil: la mejor diversión es la diversión difícil, el esfuerzo constante por superar su propia creación es parte de la tarea diaria del aprendiz.
4) Aprender a aprender: nadie puede enseñarle al otro todo lo que este necesita saber, cada persona debe hacerse cargo de su propio aprendizaje.
5) Tomarse tiempo: para hacer tareas importantes es necesario que el sujeto aprenda a manejar su propio tiempo. El apurarse o acotar el trabajo a tiempos estrechos conspira contra el aprendizaje.
6) No se pueden hacer las cosas bien sin antes haberlas hecho mal: es importante ver qué sucedió cuando algo salió mal. La libertad para equivocarse es un camino al éxito, dado que raras veces las cosas buenas y significativas salen bien al primer intento.
7) Los maestros deben hacer antes ellos mismos lo que quieren que hagan sus alumnos.
8) Entender cómo funciona el mundo digital es casi tan importante como saber leer y escribir.

En el esquema más básico, Papert propone un modelo educativo orientado a "aprender a aprender" y considera que una computadora es un objeto de motivación e incentivación esencial para su propuesta, la cual les permite a los

estudiantes abrirse a un conocimiento ilimitado, potenciando su creatividad y capacidad para resolver problemas (Veiga, 2010). La idea de un niño constructor de sus propios conocimientos fue innovadora en su época. A partir de ella se generaron una serie de emprendimientos de construcción de materiales y recursos educativos para que los niños puedan aprender sobre la base de las propuestas de Papert.

La democratización de ciertas herramientas y saberes de la mano de las tecnologías de fabricación digital abren la oportunidad para experimentar y planificar su integración con los sistemas educativos, dado que ya desde la teoría constructivista se ha indicado su potencial sobre los aprendizajes de los jóvenes. Estas condiciones materiales y sociales permiten una pedagogía fuertemente basada en la experiencia y en los procesos de desarrollo y materialización de ideas poderosas.

En el libro *Invent to Learn* (2013), Martinez y Stager indican que actividades tales como fabricación, cacharreo e ingeniería representan formas de construir saberes que se deben adoptar en las aulas, donde la fabricación hace referencia a la importancia de los procesos de construcción en el aprendizaje. Se argumenta que al realizar un trabajo con herramientas y materiales el constructor enriquece su comprensión dado que tiene un producto en mente. Del mismo modo, a las actividades de cacharreo se las percibe como una disposición mental, una forma lúdica de abordar problemas y resolverlos mediante la exploración, el descubrimiento, la experimentación, y la prueba y error. Cuando se realizan actividades de ingeniería se promueve la experiencia directa a partir de bases científicas contextualizadas y aplicadas para mejorar el mundo en el que vivimos.

Puede percibirse que el movimiento *maker* tiene, desde sus prácticas y virtudes, una relación alta con el construccionismo, ya que los participantes realizan actividades que les son personalmente significativas y suceden fuera de sus cabezas, que se materializan en objetos construidos colectivamente y compartidos. El aprender haciendo es parte esencial de los objetivos de estos colectivos. En los ambientes *maker* no solo se enseña a las personas cómo se crean y se hacen las cosas, sino que también se procura aumentar su aprecio por el mundo donde están insertos y se muestra cómo contribuir a su mejora mediante la búsqueda de soluciones creativas a ciertos problemas (por ejemplo, modificando un artefacto para que tenga una nueva funcionalidad, muchas veces reciclando elementos de la vida diaria). El movimiento *maker* procura que los estudiantes, en lugar de leer o recibir las respuestas, sean capaces de encontrarlas ellos mismos a través de procesos de creación.

Otros profesores, tales como Pérez García (2014), van un poco más allá en su perspectiva e indican que "nuestros estudiantes deben ser *hackers*", viendo a los *hackers* como personas que resuelven problemas y construyen cosas en un ambiente de aprendizaje donde reina la libertad y la colaboración. La predisposición de "manos a la obra" es lo que caracteriza a los *hackers*, es una disposición a conocer, a mejorar y a compartir. Himanen (2002) asegura que los niños son *hackers* en esencia, no hay que explicarles qué es ser *hacker*,

solamente hay que darles condiciones favorables para que desarrollen sus talentos haciéndose preguntas, buscando respuestas y formulando ideas, para luego usar todos los recursos que están a su disposición en proyectos de desarrollo. La educación es un tema clave, dado que allí se reproducen los valores culturales. Desde esta perspectiva lo más importante es ayudar a la gente a descubrir sus pasiones, aquellas donde ponen en juego su creatividad, y poder desarrollarlas. El movimiento *maker* pone el énfasis en aprender haciendo y esto está en relación directa con el espíritu *hacker*. En el colectivo la forma de aprender es por demanda y no por oferta, justo al revés que en el sistema educativo tradicional; en vez de sentarse, estar atento y escuchar, se pasa a una cultura basada en el hacer, donde se busca promover el uso crítico (no pasivo) de la tecnología y donde nuestro aporte al mundo pasa por la posibilidad de expresarnos, de crear.

Ahora más que nunca, bajo el panorama descripto, la innovación está en manos de las personas (y ya no solo de las empresas). Las prácticas que la habilitan son actividades colaborativas que dan pie a un círculo virtuoso derivado de un proceso continuo de retroalimentación. La cultura *maker* en las escuelas es algo que aún se debe explorar y experimentar con mayor intensidad, dado que potencialmente podría generar prácticas educativas enriquecedoras que inspiren, promuevan y desarrollen la creatividad y la innovación entre nuestros profesores y estudiantes.

2.5. HACIA LA FLUIDEZ TECNOLÓGICA

La interactividad puede darse desde decisiones propias, que surgen de las necesidades del usuario y son guiadas por su fluidez técnica, que facilita la inmersión en el espacio digital; o desde decisiones externas al usuario, dadas por recorridos armados por terceros (en general, con fines de consumo de bienes y servicios), donde el usuario lo único que hace es seguir caminos predefinidos. Es importante que el sistema educativo promueva y desarrolle saberes que hagan que la interactividad se dé por decisiones propias de las personas, ya que tal situación significa un grado importante de comprensión del mundo y autonomía de acción sobre él.

Mitchel Resnick ha indicado que la baja del costo de las tecnologías digitales ha facilitado el acceso de las personas a las pantallas. Esto se ve en la práctica, dado que su tenencia o uso excede clases sociales, países y culturas. De alguna manera, esta masificación de posesión ha reducido la brecha digital, la cual está relacionada con la tenencia de las pantallas. Pero aún no se ven los efectos de un uso apropiado, fluido, en pos de empoderar sus acciones y no solo de consumir servicios por caminos ya pactados (generalmente por las empresas proveedoras de máquinas y servicios). La fluidez en el uso de las tecnologías digitales no tiene que ver solo con saber usar las pantallas, sino con ir más allá, es decir, saber cómo construir cosas significativas con ellas (Resnick, 2001).

Hablar de usos fluidos de la tecnología o específicamente de fluidez digital implica una experiencia y compromiso más profundo de conocimientos y habilidades adquiridas; implica ir más allá de ciertos usos simples como buscar información, trabajar con un procesador de textos o una planilla de cálculo, usar un chat o enviar mensajes de texto. Se suele utilizar la analogía del aprendizaje de una lengua extranjera para tratar de explicar el concepto de fluidez digital: suponga que una persona aprende algunas palabras básicas del italiano, que le permitan entenderse en situaciones comunes de viaje. Así estará en condiciones de comprar algo, solicitar indicaciones para llegar a una dirección o incluso pedir el menú en un restaurante. Ahora, ese turista, en Roma, no estará en condiciones de establecer relaciones profundas con los ciudadanos ni con la cultura, dado que su comprensión y habla es bastante limitada, para nada fluida. Al carecer de un manejo fluido de la lengua italiana tampoco podrá leer el diario con cierta profundidad, ni entender plenamente lo que se dice en la radio o en la televisión, ni conversar largo y tendido, intercambiando opiniones personales.

Cuando relacionamos esta metáfora y la ponemos en el contexto de las computadoras pasa lo mismo, dado que poseer fluidez digital implica algunos saberes que están más allá de conocer cómo se usa, por ejemplo poder construir cosas significativas. Así, la fluidez puede considerarse como algo superador de lo utilitario y ponerse en función de promover un efecto catalizador en el proceso de aprendizaje. Resumiendo, a partir de las propias palabras de Resnick (2002), "[c]uando se aprende a leer y a escribir, se está en mejor posición para aprender muchas otras cosas. Sucede lo mismo con la fluidez digital. En los años venideros, la fluidez digital será un prerrequisito para obtener trabajos, participar significativamente en la sociedad y aprender a lo largo de toda la vida".

En una apuesta superadora al concepto de sociedad del conocimiento, Resnick propone un próximo estadio denominado la "sociedad de la creatividad", en atención a que el éxito, en un futuro cercano, no dependerá de "cuánto sabemos", sino de nuestra capacidad para pensar y actuar creativamente. Entonces, tratando de integrar los conceptos anteriores, podríamos decir que dada la penetración, uso y dependencia de las pantallas múltiples en la sociedad, debemos "valorar la fluidez computacional tanto como valoramos el leer y escribir" (Resnick, 2001).

Existen diferentes iniciativas desde el ámbito público y privado (por ejemplo, talleres Educ.ar, proyecto Scratch del MIT, proyecto Hora de Código, proyecto Programar, entre otros) que buscan incluir prácticas y experiencias que hagan un uso más profundo de las tecnologías en la educación de los jóvenes. En general estas iniciativas se basan en el uso de *software* para diseñar y crear imágenes digitales, sonidos, música, videos o cualquier contenido multimedia, o bien en experiencias que tienen que ver con la resolución de problemas y la automatización posterior de las soluciones basadas en prácticas de programación de *software*.

La posibilidad de crear objetos interactivos digitales utilizando la plataforma Arduino –o similares– aporta una perspectiva diferente, ya que escapa a

iniciativas como las citadas anteriormente, que se encuentran enmarcadas en el plano de la creación de *software*. De esta manera no se utiliza solamente el *software* como una herramienta que permite crear, sino también el *hardware* mismo. Tener la posibilidad de crear *hardware* nuevo y personalizado da la libertad necesaria para pensar y desarrollar nuevas aplicaciones y artefactos a medida, que estén creados con el fin de resolver problemas de diversa índole. Al ser autores no solo del *software* sino también del *hardware*, este último se desmitifica y se lo entiende como una herramienta creada a partir del control de elementos físicos y de su interpretación en una lógica digital para tomar decisiones.

Entender el funcionamiento del *hardware* y tener la posibilidad de realizar intervenciones con él permite tener una perspectiva más amplia sobre el funcionamiento de las tecnologías digitales. En un mundo donde muchos de los electrodomésticos y objetos cotidianos están dotados de sistemas embebidos, es importante no verlos como cajas negras cerradas y mágicas, sino percibirlos como artefactos que son el resultado de un diseño premeditado y que hacen uso de cierto *hardware* y *software* particular para funcionar. Leer estos aparatos y tener una idea aproximada de cómo funcionan contribuye a que se tomen mejores decisiones a la hora de usarlos, e incluso, a tener la posibilidad de extenderlos o expandirlos.

CAPÍTULO 3

Arduino y su mundo

3.1. ENTONCES, ¿QUÉ ES ARDUINO?

Arduino, como concepto, es un sistema electrónico de prototipado abierto, basado en *software* y *hardware* flexibles. Su versatilidad junto a un grupo amplio de componentes permite desarrollar de forma rápida (con conceptos elementales de electricidad y electrónica) objetos interactivos digitales que son excelentes oportunidades de aprendizaje en niños y jóvenes, con el objetivo de potenciar su creatividad y su capacidad para resolver problemas complejos.

Físicamente, y a fines prácticos, Arduino es simplemente una placa que contiene un microprocesador ATmega, una serie de pines de entrada y salida de propósito general para datos analógicos y digitales, y una conexión USB que permite cargarle programas y establecer una comunicación con una PC. Posee además ciertas protecciones para hacer más difícil que se dañe la placa al experimentar con ella y se conecta, a través de los mencionados pines de entrada y salida, con los diversos sensores y actuadores que permiten gran flexibilidad a la hora de crear objetos interactivos.

El proyecto Arduino surge en el año 2005, a partir de un grupo de estudiantes y profesores del Instituto de Diseño Interactivo Ivrea (Italia) que se propuso desarrollar un entorno de *hardware* bajo modalidad abierta. El motivo principal para llevar adelante el proyecto fue el diseño y desarrollo de una herramienta docente económica, actual y potente, que fuera accesible a los estudiantes y a las escuelas. Los microprocesadores de ese tipo, así como las placas de desarrollo de minicomputadoras que existían hasta ese momento, eran de licencias cerradas y de elevado costo, o bien simplemente estaban desactualizadas.

Más allá de su efectividad y prestaciones, el éxito y la popularidad de Arduino en el mundo se debieron a que se constituyó como una alternativa válida abierta frente a opciones propietarias y mucho más caras. Arduino también ha significado una revolución en sí mismo, dado que ha permitido a la gente encontrarse con una suerte de "arcilla electrónica" con la que modelar, casi sin restricciones, sus proyectos de creación de objetos digitales.

El primer prototipo de placa surgió de un equipo coordinado por el profesor Massimo Banzi, y se basaba en una placa electrónica muy simple, donde se ubicaba un microcontrolador básico junto con resistencias y otros elementos

que permitían la conexión de sensores y leds. Esta versión de placa no contaba con un lenguaje de programación asociado para desarrollar programas y fue únicamente utilizado por los desarrolladores para probar las compatibilidades y crear nuevas funcionalidades en pos de un diseño final. La primera versión de Arduino tenía comunicación con una computadora mediante una interfaz RS-232 y contaba con un microcontrolador ATmega8 de 8 bits.

Con la incorporación del estudiante Hernando Barragán se pudo desarrollar un entorno de programación denominado *wiring*. Al integrarse el estudiante David Cuartielles al equipo, se realizaron mejoras significativas a la interfaz de *hardware* de la placa. También se incorporaron microcontroladores para brindar soporte y memoria al lenguaje de programación. Otro colaborador principal fue el estudiante Tom Igoe, quien ayudó a mejorar la placa haciéndola más potente, y le agregó puertos USB.

Al tener un modelo más completo y robusto comenzaron la etapa de difusión mundial de las especificaciones de Arduino, así como la fabricación masiva de las placas. Las placas Arduino están basadas en los microcontroladores Atmega168, ATmega328, ATmega1280, ATmega8 y otros similares, y existe hoy en día una gran cantidad de variantes que definen a la gran familia Arduino.

Entre los integrantes más representativos de este grupo se encuentran el clásico Arduino UNO, que es ideal para gran cantidad de proyectos; el Arduino Mega, que contiene muchos más pines de entrada y salida y permite encarar proyectos más grandes; también está el Arduino Nano usado en robótica y en aplicaciones portátiles gracias a su pequeño tamaño; y, por último, el Lilypad, diseñado para integrarse en ropas y diversas telas.

3.2. *OPEN SOURCE* Y *OPEN HARDWARE* COMO FILOSOFÍA DE TRABAJO

Arduino, desde su concepción, pertenece al mundo del *open source* (*software* libre) y del *open hardware* (*hardware* abierto), alineado con la filosofía de trabajo que ambos movimientos proponen. Tanto el *hardware* como el *software* abierto fomentan el desarrollo de tecnologías cuyos diseños son públicos y sin secretos, con el fin de que otros hagan uso y modifiquen esa tecnología. Esto da lugar a que el uso y los costos de las tecnologías sean bajos o nulos, y se facilite el acceso a más cantidad de usuarios; del mismo modo y en sentido inverso, también fomenta que las personas puedan participar y hacer aportes a esas mismas tecnologías, ayudándolas a madurar y mejorar.

Cuando se habla de *open hardware*, se hace referencia al *hardware* con diseño público, es decir, liberado para que cualquier persona pueda estudiarlo, modificarlo y distribuirlo, así como producir y vender *hardware* basado en ese diseño.

El movimiento de *hardware* libre tiene por finalidad construir una gran librería abierta y accesible para todo el mundo, en beneficio de un ahorro significativo de dinero y esfuerzo humano en diseños electrónicos redundantes.

De alguna manera, el *hardware* en modalidad abierta implica poder ver su interior, abrir los objetos y ver cómo funcionan realmente.

La importancia que tienen estos movimientos no es menor, ya que demuestran que el trabajo autoorganizado, público y comunitario es capaz de crear tecnología competitiva, robusta e innovadora; y en este mismo sentido plantea una pregunta interesante: si esta filosofía hace posible crear tecnología de la denominada "de punta", ¿qué más es posible de hacer con esta metodología de trabajo colaborativa, abierta y autoorganizada?

Bajo este supuesto resulta interesante hacer que los jóvenes tomen conocimiento y participen en este tipo de iniciativas. Esto hace que Arduino sea más adecuado aún para el trabajo en el aula que otras alternativas de placas similares más cerradas.

Más allá de desarrollar las habilidades técnicas, fomentar y cultivar la creatividad, y promover capacidades y aptitudes en la resolución de problemas interdisciplinarios, experimentar con la plataforma Arduino permite estar en contacto con grupos colaborativos de gente que crea, mantiene y comparte sus proyectos y conocimientos en una forma de trabajo desinteresada y con vistas a producir nuevos conocimientos y tecnologías que pueda usar el resto de la sociedad.

3.3. COMUNIDADES ALREDEDOR DE ARDUINO

Debido a la filosofía de trabajo y al nicho al que apunta Arduino, existen alrededor de él y en distintas partes del mundo muchísimas comunidades *online* dedicadas a descubrir y compartir usos de esta tecnología:desde los foros oficiales de Arduino en http://forum.arduino.cc/, a sitios de *makers* y comunidades DIY como http://www.instructables.com/, foros de electrónica (http://www.forosdeelectronica.com) o incluso comunidades específicas de uso de esta placa (http://clubarduino.com.ar). Muchos de estos ejemplos están en español y surgen desde diversas perspectivas.

En estas comunidades prima el compartir, ya sea una duda, una producción, una idea, una propuesta o una preocupación; y aquellos participantes que más aportes hacen son los más valorados por las comunidades. Cada aporte hecho en estos sitios tiene un valor técnico pero, además, un valor social muy grande. El espíritu de estos colectivos se basa en ayudar y recibir ayuda de otros para lograr crear proyectos; forma grupos sociales que comparten intereses y vivencias.

Estas comunidades representan un espacio interesante desde el punto de vista de la integración de Arduino en las aulas ya que, por un lado, dan la posibilidad de recibir asistencia técnica y guía en la concreción de proyectos, pero por otro, muestran un esquema alternativo de aprendizaje informal que está basado en valores sociales muy fuertes que tienen que ver con compartir, con hacer y con ayudar.

3.4. ARDUINO EN AMBIENTES EDUCATIVOS

Es importante considerar que para poder sacar provecho de la introducción de dichas tecnologías en las aulas y potenciar los aprendizajes de los estudiantes, no se debe intentar solamente integrar a la placa Arduino como un contenido más de clase de alguna materia, tales como informática o tecnología. Por el contrario, el verdadero aporte que pueden hacer estas herramientas en las aulas es nutrir las experiencias de enseñanza y aprendizaje escolares a partir de prácticas propias del aprendizaje informal.

Trabajar a través de proyectos que surjan de los intereses de los alumnos puede ser una buena forma de iniciar este recorrido. Usar objetos o problemáticas que sean parte de sus intereses representa un gran elemento de carácter motivador, pues permite que los alumnos sientan afinidad hacia los proyectos y los tomen como algo personal, algo de lo cual son dueños.

Dejar que los alumnos se enfrenten a preguntas sin respuestas previamente definidas (que puede desconocer incluso el docente) presenta a estos proyectos como verdaderos desafíos no controlados. En parte, el rol docente sufre transformaciones gracias al desconocimiento compartido con los alumnos y a que juntos deberán enfrentarse a la búsqueda y creación del conocimiento necesario para llevar a cabo los proyectos, que han surgido de acuerdos mutuos.

Lograr la concreción de esos proyectos significa crear respuestas a múltiples problemas sin dejar que la falta de motivación, la frustración u otros agentes externos se interpongan en este camino. Asimismo, dejar que las particularidades de cada alumno tengan protagonismo significa hacer frente a variedades de ritmos y velocidades de aprendizajes, capacidades y formas de trabajo. Y estos son solo algunos de los desafíos a los cuales el docente se deberá enfrentar; sin embargo, no estará solo ya que trabajará a la par con sus alumnos. El carácter multidisciplinario de los proyectos que utilizan estas tecnologías garantiza, por otra parte, que más de un docente pueda involucrarse en los proyectos.

De igual modo, las consultas a comunidades en línea y el uso de diversas fuentes que se pueden encontrar en Internet serán de gran ayuda. Así también aprenderán a utilizar y leer críticamente la información que se puede hallar en la red.

Finalmente, estamos convencidos de que la utilización de estas herramientas digitales en las aulas tiene implicaciones sumamente enriquecedoras para los procesos de enseñanza y de aprendizaje de cualquier joven en la educación secundaria. Resumimos sus ventajas y fortalezas principales en los siguientes puntos:

- Propicia el trabajo colaborativo.
- Pone a los estudiantes en el rol de pensadores creativos.
- Aleja a los estudiantes de ser solo consumidores de tecnología, para habilitarlos como creadores de esta.
- Promueve un aprendizaje profundo desde la práctica, el cual es motivado por procesos internos al estudiante.

- Facilita una rápida adopción por parte de los estudiantes y profesores, ya que los requisitos y saberes previos para empezar a crear son pocos.
- Promueve un trabajo interdisciplinario.
- Promueve habilidades de resolución de problemas complejos.
- Ayuda al desarrollo del pensamiento computacional.
- Su pedagogía es innovadora, dado que está enmarcada en el aprendizaje activo (Korb), la espiral del pensamiento creativo (Resnick) y el construccionismo (Papert).
- El aprendizaje se da en el proceso que va desde el diseño hasta la reflexión sobre lo construido, lo experimentado, lo probado y lo discutido.
- Ayuda a formar estudiantes que, además de leer, puedan expresarse escribiendo y creando con los códigos de su tiempo.
- Lo lúdico está presente y ayuda. Es una herramienta perfecta para vocaciones artísticas, diseñadores de todo tipo y cualquiera interesado en crear entornos u objetos interactivos.

Bibliografía

Brünner, José Joaquín
2006 "Preguntas del siglo XXI", en Santander, María de los Ángeles (comp.), *Ideas para una educación de calidad*, Santiago de Chile, Fundación Libertad y Desarrollo.

Bollier, David
2010 *The Promise and Peril of Big Data*, Washington,The Aspen Institute.

Carpenter, Edmund y McLuhan, Marshall
1968 *El aula sin muros. Investigaciones sobre técnicas de comunicación*, Barcelona, Editorial Cultura Popular.

Casacuberta, David
2003 *Creación colectiva. En Internet el creador es el público*, Barcelona, Gedisa, 1ª ed.

Castells, Manuel
2001 *La Galaxia Internet*, Barcelona, Plaza & Janés.

Dougherty, Dale
2013 "The Maker Mindset", en Honey, Margaret y Kanter, David (comps.), *Design, Make, Play: Growing the Next Generation of STEM Innovators*, Nueva York, Routledge.

Evans, Dave
2011 "The Internet of Things. How the Next Evolution of the Internet is Changing Everything", Cisco Internet Business Solutions Group (IBSG). Disponible en: <https://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf> [fecha de consulta: 30 de septiembre de 2015].

Gardner, Howard
2000 *Estructuras de la mente. Teoría de las inteligencias colectivas*, México, Fondo de Cultura Económica.
2008 *Las cinco mentes del futuro*, Barcelona, Paidós.

Harel, Idit
2002 "El aprendizaje de medios nuevos. Una necesidad nueva para la generación de jóvenes *clickerati*", en MaMaMedi Inc. Disponible en:

<http://web.media.mit.edu/~calla/web_comunidad/Readings/aprendizaje.pdf> [fecha de consulta: 30 de septiembre de 2015].

Himanen, Pekka
2002 *La ética del hacker y el espíritu de la era de la información*, Barcelona, Destino.

Lanier, Jaron
2011 *Contra el rebaño digital. Un manifiesto*, Barcelona, Editorial Debate.

Le, Dustin
2015 "The Maker Movement and the Classroom", en *Edudemic*, 15 de julio. Disponible en: <http://www.edudemic.com/maker-movement-classroom/> [fecha de consulta: 30 de septiembre de 2015].

Martín-Barbero, Jesús
2003 "Saberes hoy: diseminaciones, competencias y transversalidades", en *Revista Iberoamericana de Educación* (OEI), nº 32, Madrid, mayo-agosto.

Martinez, Silvia Libow y Stager, Gary
2013 *Invent to Learn. Making, Tinkering, and Engineering in the Classroom*, Torrance (CA),Constructing Modern Knowledge Press.

Papert, Seymour
1996 *The Connected Family: Bridging the Digital Generation Gap*, Atlanta (GA), Longstreet Press.

Pérez García, Francisco
2014 "Creatividad tecnológica mediante programación", en *Revista Didáctica, Innovación y Multimedia*, nº 30.

Resnick, Mitchel
2001 "Closing the Fluency Gap", en *Communications of the ACM*, vol. 44, nº3, marzo.
2002 "Rethinking Learning in the Digital Age" [Repensando el aprendizaje en la era digital], en Kirkman, Geoffrey *et al.*, *The Global Information Technology Report 2001-2002. Readiness for the Networked World*, Nueva York, Oxford University Press.
2009 *Sembrando semillas para una sociedad más creativa*, sitio web Eduteka. Disponible en: <http://www.eduteka.org/modulos.php?catx=9&idSubX=277&ida=914&art=1> [fecha de consulta: 30 de septiembre de 2015].

Stager, Gary
2014a "Progressive Education and The Maker Movement–Symbiosis or Mutually Assured Destruction?". Disponible en: <http://www.inventtolearn.com/wp-content/uploads/2014/10/FabLearn-2014-paper-for-web1.pdf> [fecha de consulta: 30 de septiembre de 2015].
2014b "What's the Maker Movement and Why Should I Care?", sitio web Scholastic. Disponible en: <http://www.scholastic.com/browse/article.jsp?id=3758336> [fecha de consulta: 30 de septiembre de 2015].

Stephens, Dale J.
2013 *Hacking Your Education. Ditch the Lectures, Save Tens of Thousands, and Learn More than Your Peers Ever Will*, Londres, Penguin.

Tirado Fernández, Elena
2015 "IoT: una revolución industrial y en la relación con el consumidor", sitio A un clic de las TIC. Disponible en: <http://www.aunclicdelastic.com/iot-una-revolucion-industrial-y-en-la-relacion-con-el-consumidor/> [fecha de consulta: 30 de septiembre de 2015].

Veiga, Leonardo
2010 "Es el Plan Ceibal y el corporativismo en la educación", en *Revista de Antiguos Alumnos del IEEM*, febrero. Disponible en: <http://socrates.ieem.edu.uy/wp-content/uploads/2011/05/es-el-plan-ceibal_veiga.pdf> [fecha de consulta: 30 de septiembre de 2015].

Segunda parte
Prácticas con Arduino

Acerca de los proyectos

A continuación se presentan una serie de proyectos desarrollados utilizando la tecnología Arduino, con los que pretendemos iniciar el camino hacia la construcción de artefactos digitales interactivos. Cada uno de estos proyectos aborda en forma simultánea contenidos de electrónica y programación aplicados al entorno de la placa Arduino, por lo que no se requieren conocimientos previos en ninguna de estas áreas para comenzar a trabajar.

Aunque no es obligatorio, los proyectos fueron pensados para que se realicen en el orden en el que son expuestos, ya que van aumentando en complejidad y presentan de forma escalonada los saberes pertinentes para crear objetos digitales interactivos cada vez más completos. Este material ofrece un punto de partida y un andamiaje en los tres frentes que se requieren para construir esta clase de artefactos (electrónica, programación y sistemas embebidos), pero no aspira a ser un curso completo en la temática ni una guía "paso a paso" cerrada en sí misma.

Su verdadero objetivo es que funcione como una invitación a entender mejor las tecnologías, a experimentar con ellas, a intervenirlas y –sobre todo– a constituirnos como participantes activos de un proceso de creación y expresión a través de ellas.

Preparar el entorno: descargar e instalar Arduino

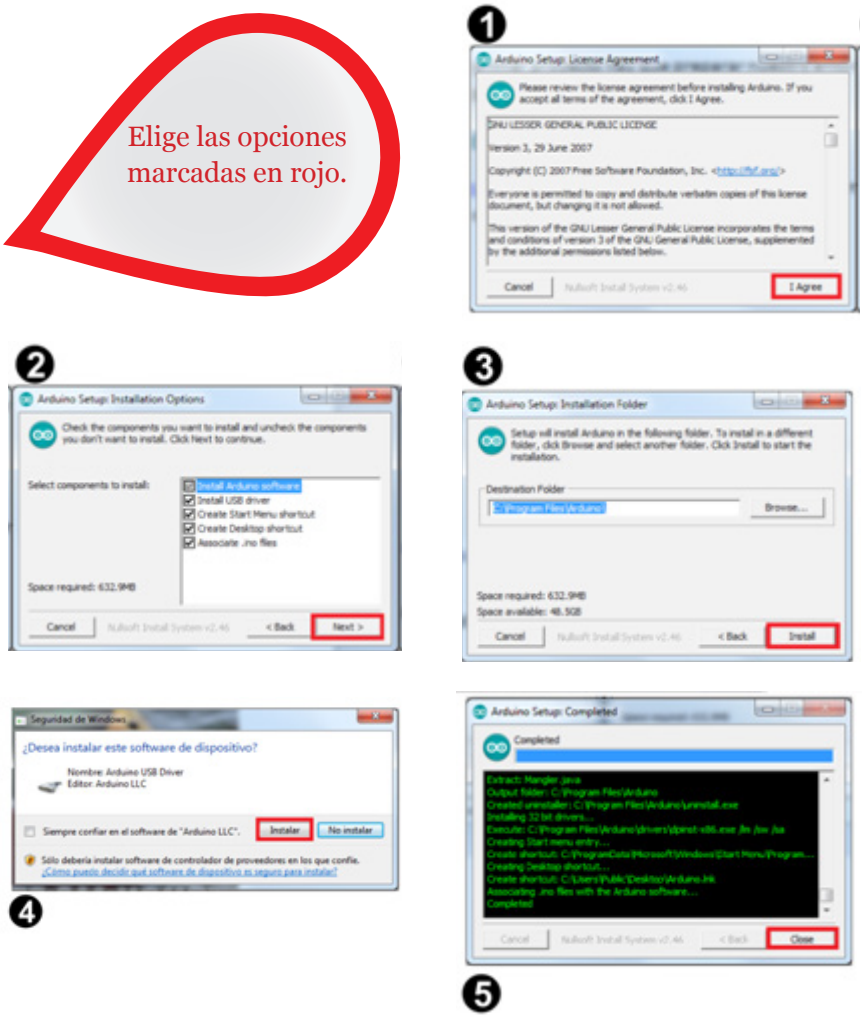
Antes de comenzar con el primer proyecto tenemos que preparar nuestro entorno de trabajo y lo vamos a hacer en cuatro simples pasos.

1) DESCARGA EL ENTORNO ARDUINO

Ingresa a esta página de Arduino (http://arduino.cc/en/Main/Software) y elije el sistema operativo donde lo vas a utilizar. Por ejemplo, si lo instalarás en Windows:

2) INSTALA EL ENTORNO ARDUINO

Las siguientes imágenes te van a servir para guiarte por el proceso de instalación. Es posible que en el momento en que instales el IDE (entorno de desarrollo) haya una versión nueva disponible, pero esencialmente los pasos serán los mismos.

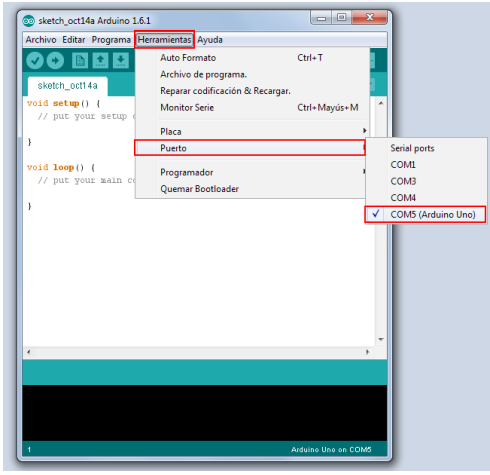


3. CONECTA TU ARDUINO

Usando el cable USB conecta tu Arduino a la computadora.

4. EJECUTA EL IDE

Busca en el escritorio el acceso directo al IDE Arduino  y dale doble clic. Luego verifica que en Herramientas -> Puerto se encuentre seleccionado el puerto COM que tiene conectado el Arduino. Aparecerá entre paréntesis “Arduino Uno”, como muestra la imagen:

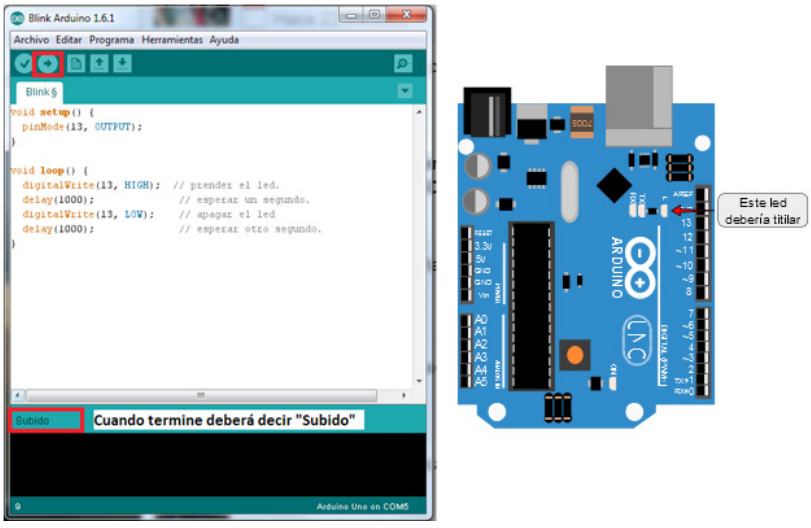


Por último, copia y pega este código en la pantalla principal.

Código: PruebaCero

```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH);  // prender el led.
  delay(1000);               // esperar un segundo.
  digitalWrite(13, LOW);    // apagar el led
  delay(1000);               // esperar otro segundo.
}
```

Luego, cárgalo en el Arduino haciendo clic en el ícono, como muestra la siguiente imagen:

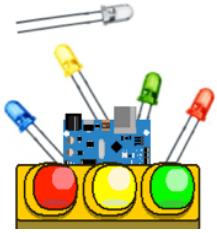


Si todo salió bien, el led marcado en la imagen debería titilar. ¡Ahora si estás en condiciones de trabajar con Arduino! Y, por qué no, empezar con los proyectos.

El semáforo: jugar con luces y colores

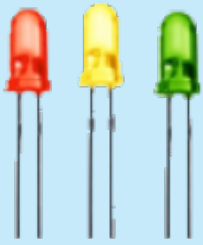
SOBRE ESTE PROYECTO

Nuestro primer experimento será nada más y nada menos que un semáforo. Si bien no tendrá el tamaño de un semáforo real, funcionará de la misma manera.



Tags: leds, digital, resistencias, semáforo, loop, setup.

Elementos necesarios



3 LEDS (ROJO, AMARILLO, VERDE)

Los leds son un tipo especial de diodos que emiten luz cuando la corriente eléctrica circula por ellos.

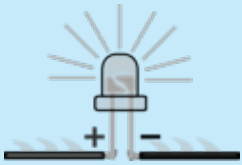
3 RESISTENCIAS 220

Las resistencias son un componente electrónico que actúa como una barrera y deja pasar solo cierta cantidad de electricidad (medida en Ohmios).

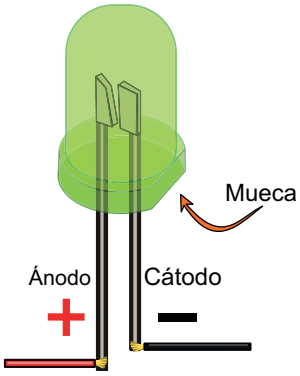
MANOS A LA OBRA

Un semáforo no es más que una señal lumínica que le comunica a los conductores qué es lo que deberían hacer: si deben esperar o avanzar. Para crear nuestro mini semáforo, por lo tanto, vamos a necesitar luces que indiquen el estado del mismo y además algo que maneje los tiempos en que las luces van a estar prendidas. Las luces que utilizaremos van a ser leds y ese algo que va a controlarlas será nuestro pequeño programa en el Arduino.

i **Sobre los leds** Conociendo un poco más

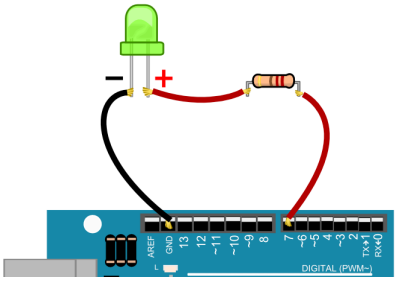


Los **leds** pertenecen a un grupo de componentes electrónicos llamados diodos, que se comportan diferente dependiendo de la dirección en que circula la corriente eléctrica que los atraviesa. Por eso es importante conectarlos bien, ya que se pueden dañar.



Al conectar los leds tendrás que tener un especial cuidado ya que las patitas de los mismos deben estar correctamente conectadas, tal como muestra la figura de la derecha.

La patita que corresponde a la mueca o marca que tiene el led en uno de sus lados corresponde al polo negativo y va conectado al pin ground en nuestro Arduino.



Siguiendo el esquema de la izquierda vamos a armar la primera prueba. Conectaremos la salida digital 7 a una resistencia de 220ohm y esta resistencia a su vez a la pata positiva del led verde. Luego, conectaremos la pata negativa (o cátodo) al pin etiquetado como ground (GND) del Arduino, para cerrar el circuito.

El uso de la resistencia es necesario ya que la placa Arduino envía más corriente eléctrica de la que el led puede manejar (esto es si se encuentra encendido por un tiempo prolongado).

Ahora sí, una vez que este circuito básico está armado, estamos listos para darle las primeras instrucciones a nuestro Arduino. Para eso, crea en el entorno Arduino un nuevo sketch (programa) y escribe lo siguiente:

Código: PruebaLeds_01

```
// PruebaLeds_01
void setup() {
        pinMode(7,OUTPUT); // le decimos a Arduino que usaremos este pin como salida.
        digitalWrite(7,HIGH); // enviar la señal HIGH al pin 7 (encender el led).
}
void loop() {
        // por ahora no vamos a hacer nada acá.
}
```

Compílalo, cárgalo y pruébalo. ¿Qué es lo que pasó?

✓Si todo salió bien	El led se encendió y no volvió a apagarse.
×Si algo salió mal	El led no se encendió.
Qué revisar si no funcionó como debía...	✓ Revisar que el conexionado sea el correcto. ✓ Revisar que no se haya soltado ningún cable. ✓ Revisar que el código esté correcto y no falten ";" ni llaves por cerrar. (Haga clic en el bóton dentro del entorno Arduino para verificar el código). ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar que el led esté correctamente conectado: el ánodo al pin 7 y el cátodo al pin ground (Tierra).

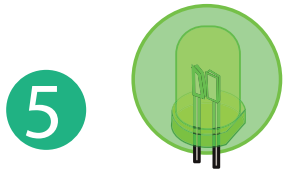
¡Perfecto! No es el mejor semáforo del mundo, ya que deja que todos pasen sin control, pero recién es nuestra primera prueba. Vamos a repasar con detenimiento qué es lo que sucedió:

El código escrito en el entorno de programación se ha compilado (es decir traducido para que el procesador lo pueda entender).

El programa se cargó en el Arduino a través de la comunicación establecida por la conexión USB.

El programa cargado en el Arduino se ejecuta y se escribe "High" en el pin 7.

Al escribir en el pin la señal HIGH, la corriente comienza a circular por nuestro cirtuito desde el pin 7 hasta el pin Ground del Arduino.



Cuando la corriente circula por nuestro flamante led, ¡este se enciende!

Sobre los pines, las señales digitales y analógicas...

El Arduino dispone de varios pines que permiten conectar salidas y entradas digitales o analógicas. Ya veremos la diferencia entre ellas más adelante, por ahora basta decir que las señales digitales solo pueden tener dos valores posibles, mientras que las analógicas pueden tener muchos valores. Los pines digitales son los que se encuentran etiquetados del 0 al 13, como se ve en la figura siguiente:

Que sean digitales significa que en estos pines solo puede haber dos valores posibles, un voltaje "Alto" (HIGH) o un voltaje "Bajo" ("LOW"). Estos pines se pueden usar tanto para escribir, es decir para producir una salida (prender un led en nuestro caso), como para leer una entrada (la que puede proveer un sensor).

Para prender el led podemos utilizar esta línea de código:

Fragmento de código

```
digitalWrite(7,HIGH); // enviar la señal HIGH al pin 7 (encender el led)
```

Y para apagarlo esta otra:

Fragmento de código

```
digitalWrite(7,LOW); // enviar la señal LOW al pin 7 (apagar el led)
```

Con esta nueva información ya estamos preparados para prender y apagar el led, que es para lo que podríamos utilizar un código como el siguiente:

Código: PruebaLeds_02

```
// PruebaLeds_02
void setup() {
        pinMode(7,OUTPUT); // le decimos a Arduino que usaremos este pin como salida
        digitalWrite(7,HIGH); // enviar la señal HIGH al pin 7 (encender el led)
        digitalWrite(7,LOW); // apagar el led
}
void loop() {
         // por ahora no vamos a hacer nada acá
}
```

Compílalo, cárgalo y pruébalo. ¿Qué es lo que pasó?

✓Si todo salió bien	El led se encendió e inmediatamente se apagó (es probable que ni siquiera lo hayas visto encendido).
×Si algo salió mal	El led sigue encendido.
 Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB.

Como el Arduino es tan rápido (bueno, en realidad muy lento comparado con una PC de escritorio o incluso con un celular), apenas escribió HIGH en el pin 7, inmediatamente escribió LOW en el mismo pin, haciendo que el led se prenda por una pequeñísima fracción imperceptible de tiempo y se apague.

Lo que necesitamos hacer es que el Arduino espere un momento antes de apagar el led, para dejarnos apreciar lo que sucede. Para indicarle al Arduino que espere un tiempo antes de ejecutar la siguiente acción, podemos usar el siguiente código:

Fragmento de código

```
delay(1000); // esperar 1000 milisegundos.
```

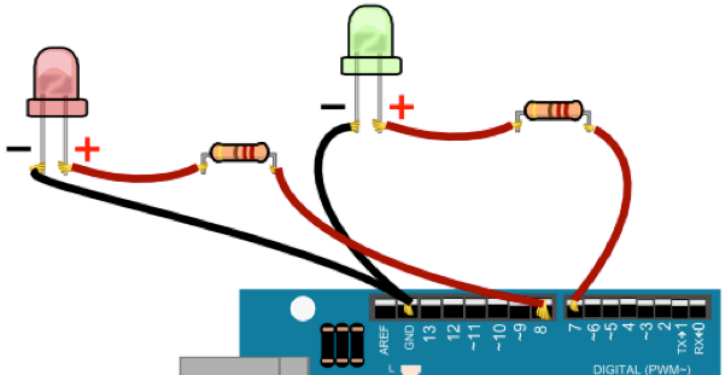
Vamos a ver el ejemplo completo de cómo indicarle al Arduino que deje el led encendido un tiempo para poder verlo. Copia, pega y carga el siguiente código en tu Arduino:

Fragmento de código

```
// PruebaLeds_03
void setup() {
        pinMode(7,OUTPUT); // le decimos a Arduino que usaremos este pin como salida
        digitalWrite(7,HIGH); // enviar la señal HIGH al pin 7 (encender el led)
        delay(5000); // esperar 5 segundos
        digitalWrite(7,LOW); // apagar el led
        delay(1000); // esperar 1 segundo
        digitalWrite(7,HIGH); // volver a encenderlo
}

void loop() {
        // por ahora no vamos a hacer nada acá
}
```

✓Si todo salió bien	El led se encendió, pasaron 5 segundos y se apagó. Luego pasó un segundo y volvió a encenderse.
×Si algo salió mal	El led no prendió.
 Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB.

Muy bien, ahora sí, para completar nuestro semáforo sólo debemos agregar un led más, siguiendo el siguiente esquema:



Ahora ya tenemos dos leds para empezar a jugar. Vamos a hacer que primero prenda el verde mientras el rojo esté apagado, y luego al revés, utilizando el siguiente código:

Código: PruebaLeds_05

```
// PruebaLeds_05
void setup() {
        pinMode(7,OUTPUT); //  usamos el pin 7 como salida (led verde)
        pinMode(8,OUTPUT); //  usamos el pin 8 como salida (led rojo)
        digitalWrite(8,HIGH); // enviar la señal HIGH para prender el led rojo
        digitalWrite(7, LOW); // enviar la señal LOW para asegurarnos de que el verde este apagado
        delay(5000); // esperar 5 segundos
        digitalWrite(8,LOW);  // apagamos el led rojo
        digitalWrite(7, HIGH); // encendemos el verde
        delay(5000); // esperar 5 segundos
        // hacemos todo una vez más
        digitalWrite(8,HIGH); // enviar la señal HIGH para prender el led rojo
        digitalWrite(7, LOW); // enviar la señal LOW para asegurarnos de que el verde este apagado
        delay(5000); // Esperar 5 segundos
        digitalWrite(8,LOW);  // apagamos el led rojo
        digitalWrite(7, HIGH); // encendemos el verde
}

void loop() {
        // por ahora no vamos a hacer nada acá
}
```

✓Si todo salió bien	El led verde empezó apagado y el rojo encendido. Cinco segundos después, el verde se prendió y el rojo se apagó. Cinco segundos después, volvió a suceder lo mismo.
×Si algo salió mal	Los leds no se prendieron en el orden correcto o no se prendieron en absoluto.

Qué revisar si no funcionó como debía...

- ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB.
- ✓ Revisar el conexionado: comprobar que ambos leds tengan conexión al pin ground y que el ánodo y el cátodo de cada uno estén correctamente conectados.

Ahora la pregunta que puede surgir es... ¿cómo hacemos para que el semáforo siga funcionando sin tener que agregar el código para hacer simplemente lo mismo?

Como puedes ver, todos los códigos que enviamos a Arduino tienen dos estructuras, una llamada "setup" y otra llamada "loop":

Fragmento de código

```
void setup() {}
void loop() {}
```

Estas estructuras son llamadas funciones y dentro de ellas se escribe código (luego volveremos sobre el concepto de función). Lo que escribimos dentro de la función setup() se ejecuta solo una vez cuando el Arduino se enciende. Mientras que la función loop() se ejecuta una y otra vez mientras el Arduino tiene corriente.

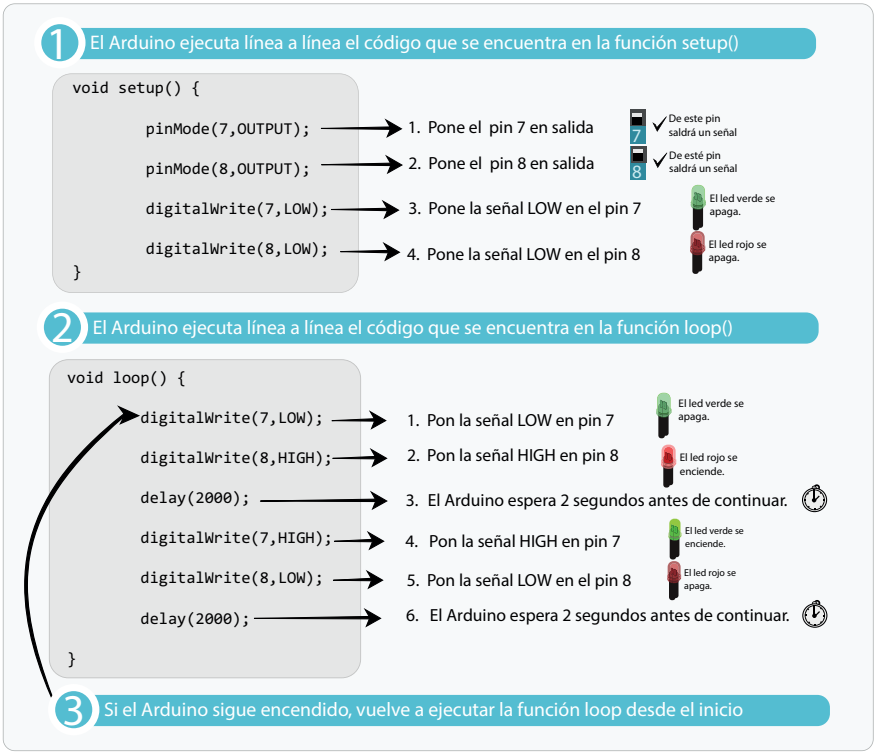
Finalmente, para que nuestro semáforo quede terminado, este sería un posible código final:

Código: semáforo

```
// Semaforo
void setup() {
    // lo que está dentro de setup se ejecuta solo una vez.
         //  solo necesitamos decirle una vez al Arduino cómo usaremos los pines 7 y 8
       pinMode(7,OUTPUT); // verde
       pinMode(8,OUTPUT); // rojo
       // comenzamos con ambos apagados
       digitalWrite(7,LOW); //
       digitalWrite(8, LOW); //

}
void loop() {
// este código se va a ejecutar una y otra vez
       digitalWrite(7,LOW); //  apagamos el verde
       digitalWrite(8,HIGH); //  encendemos el rojo
       delay(2000); // esperamos 2 segundos
       digitalWrite(7,HIGH); //  encendemos el verde
       digitalWrite(8,LOW); //  apagamos el rojo
       delay(2000); // esperamos 2 segundos
       // una vez que termina la función loop, se vuelve a ejecutar desde el comienzo
}
```

✓Si todo salió bien	Nuestro semáforo funciona. Primero se enciende el rojo, luego se apaga y se prende el verde, ¡y así para siempre! (o hasta que le cortes la corriente).
×Si algo salió mal	Los leds no se prendieron en el orden correcto o no se prendieron en absoluto.
Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado: comprobar que ambos leds tengan conexión al pin ground y que el ánodo y cátodo de cada uno estén correctamente conectados

Vamos a repasar qué es lo que sucedió cuando cargamos este código al Arduino:

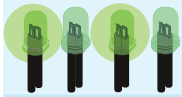


¡Muy bien, ahora sí hemos completado el semáforo! A continuación hay una serie de experimentos que puedes hacer antes de dar por terminado este proyecto.

¡HORA DE EXPERIMENTOS Y EJERCICIOS!

Ahora que ya sabes un poco más sobre Arduino, por qué no intentás...

1. Haz que el semáforo pase más tiempo en rojo que en verde.



2. Haz que el led verde empiece a titilar antes de que se apague y se encienda el rojo, tal como los semáforos actuales.

3. Agrega un led amarillo que se encienda entre medio del rojo y el verde. Ahora sí estaría completo.

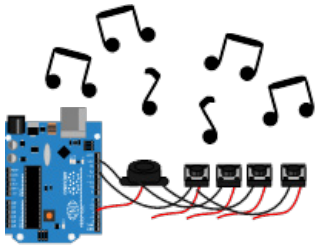
4. ¿Se te ocurre como hacer para que el semáforo, al encenderse, prenda los tres leds por unos segundos y luego funcione normalmente? (*pista*: recordá la diferencia entre setup y loop)

PROYECTO 2

Minipiano: haciendo música con Arduino

SOBRE ESTE PROYECTO

En este segundo experimento jugaremos con botones y con un buzzer para entender cómo podemos hacer para leer datos y utilizarlo en los programas que hacemos en Arduino. ¡Además vas a crear tu propio instrumento electrónico!



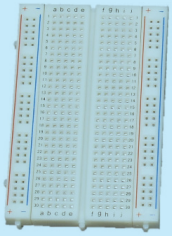
Tags: digital, resistencias, condicionales, if, botones, protoboard.

Elementos necesarios

BUZZER
Los buzzers son capaces de emitir sonidos porque en su interior cuentan con un electroimán que hace vibrar una placa de metal.

3 RESISTENCIAS 220Ω
Las resistencias son un componente electrónico que actúa como una barrera y deja pasar solo cierta cantidad de electricidad (medida en Ohmios).

3 BOTONES
Los botones simplemente unen los extremos de sus patas al ser presionados para permitir el paso de la corriente eléctrica.

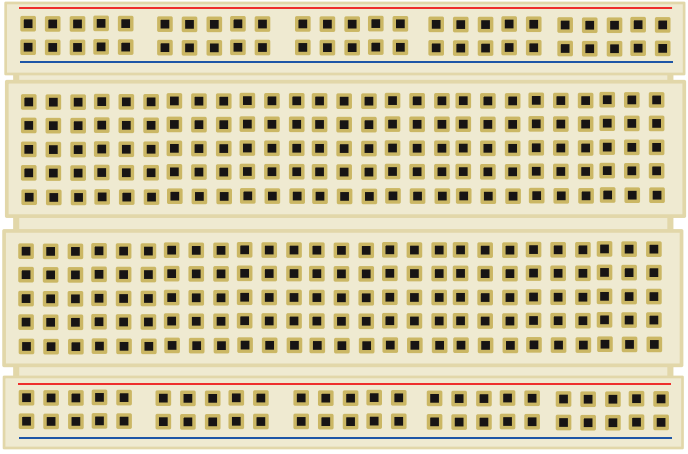


PROTOBOARD
Es un lugar donde podemos conectar los cables y todos los componentes de nuestro proyecto de forma prolija y fácil.

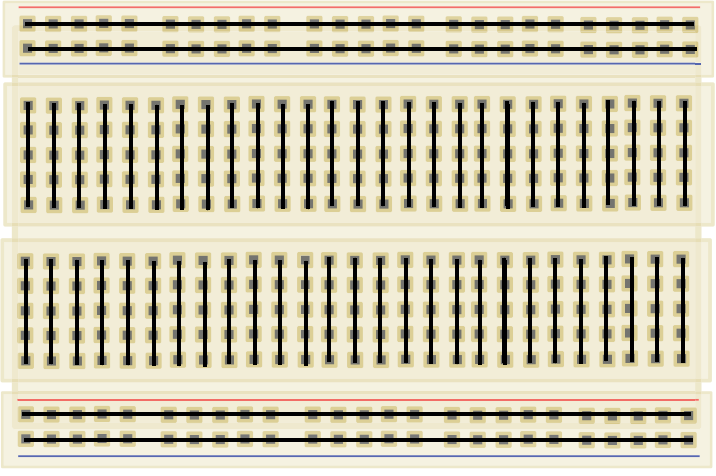
MANOS A LA OBRA

En este proyecto vamos a hacer un pequeñísimo piano de 3 teclas y para ello vamos a usar nuevos elementos: el protoboard, los botones y un buzzer.

El protoboard es un elemento que vamos a utilizar mucho en nuestros proyectos con Arduino y en general con cualquier otro proyecto que tenga que ver con electrónica. Como pudiste notar en el proyecto anterior, tener los cables sueltos y enredados hace muy complicada la conexión de los componentes.



El protoboard es una solución perfecta a este problema, ya que permite conectarlos de manera que no se muevan y sin la necesidad de realizar soldaduras. La placa consiste simplemente en un plástico con pequeños agujeros que están unidos por conexiones internas. En la siguiente figura puedes ver cómo están conectados los pines de la placa.



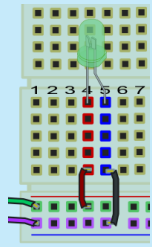
La parte inferior y superior están unidas de por lineas horizontales. Es aquí donde usualmente se conectan la alimentación en rojo y la tierra en la azul.

Por otro lado, los componentes electrónicos se suelen conectar en el sector del medio, donde las conexiones son en líneas verticales.

En este ejemplo, al conectar el led, toda la columna 4 de ese sector queda unida al ánodo del led, y la columna 5 al cátodo.

Basta con conectar ahora por ejemplo un cable a uno de los pines de la columna 5 para poder conectar algo al ánodo del led.

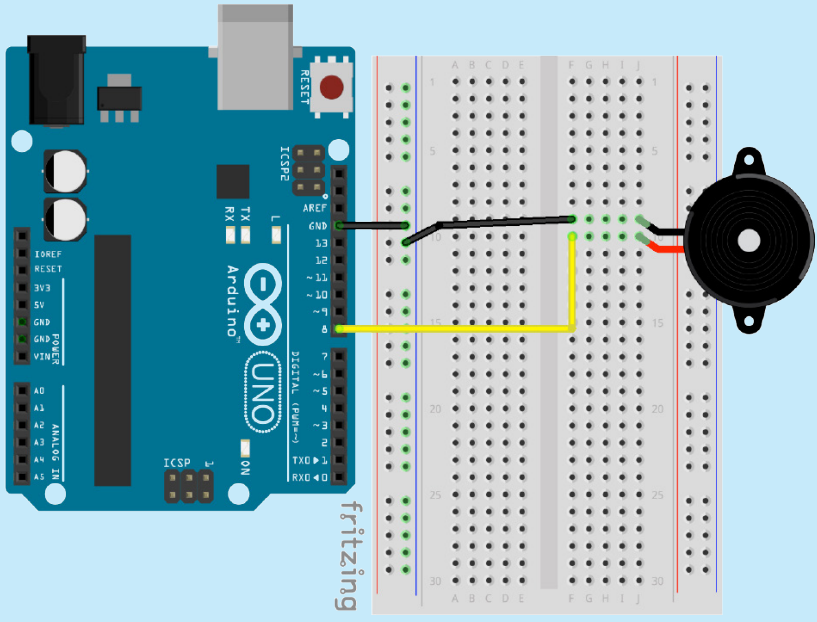
Las filas inferiores, que corren en horizontal, están interconectadas.



Los primeros sonidos

Antes de comenzar a armar nuestro extenso piano de tres teclas, vamos a ver cómo hacemos para lograr que nuestro Arduino emita un simple ruido. Haz el siguiente conexionado con el Arduino, usando el protoboard:

Esquema:HaciendoRuido



Luego escribe el siguiente código y cárgalo en tu Arduino.

Código: HaciendoRuido

```
// Haciendo Ruido
void setup() {
    // le decimos al Arduino que usarmos el pin 8 como salida
    pinMode(8,OUTPUT);
}
void loop() {
    // usamos la función tone, que emite un sonido en el pin 8, con la frecuencia 264
    tone(8,264);
    // Esperamos 1 segundo
    delay(1000);
    // emitimos otro sonido
    tone(8,294);
    delay(1000);
    // dejamos de emitir sonido
    noTone(8);
    delay(1000);
}
```

¡Ahora pruébalo!

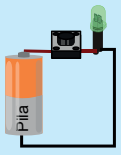
✓Si todo salió bien	El invento reproducirá dos tonos de un segundo de duración y esperará otro segundo para volver a repetirse, una y otra vez.
×Si algo salió mal	No se escucha nada.
Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado: comprobar que cada una de las patas del buzzer esté conectada, una al ground y otra al pin 8.

¡Genial! Ya no estamos en silencio, si quieres puedes hacer que tu Arduino reproduzca una canción usando código. También puedes experimentar qué es lo que pasa con el sonido cuando la frecuencia es más alta o más baja.

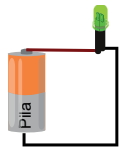
El primer botón

Llegó el momento de aprender a usar los botones para poder darle órdenes a nuestro Arduino. Lo que hace un botón es, simplemente, unir o interrumpir el paso de la corriente eléctrica.

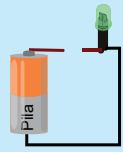
Imagina el siguiente circuito donde un botón (luego veremos cómo) está conectado a una pila y a un led.



Cuando el botón está presionado, lo que hace es unir los extremos, permitiendo el paso de la corriente eléctrica por el led

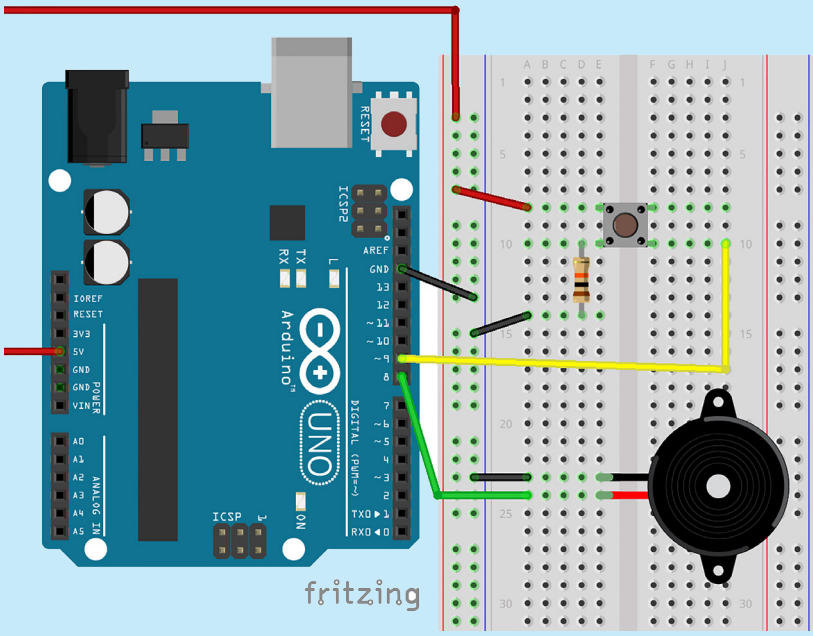


Cuando el botón no está siendo presionado es como si no existiese y los cables simplemente no estuvieran conectados.



Ahora que entendemos cómo funciona un botón, vamos a ver cómo conectarlo al Arduino para que al presionar el botón podamos leer en el pin 9 el estado del mismo.

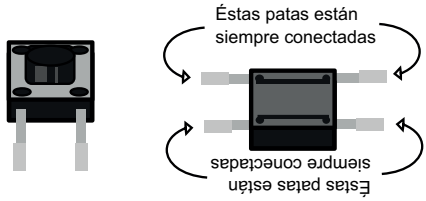
Esquema: conectando un botón al minipiano



El esquema de conexión se ve más complicado de lo que en realidad es y para comprenderlo solo restan saber dos cosas:

- Los pares de patas del botón están siempre conectados. Por lo que el pin que conecta el botón al ground, a la resistencia y al pin 9 están siempre todos conectados.

- Lo que sucede al presionarse el bóton es que fluye la corriente eléctrica y llega al ground y al pin 9.



Para poder leer un valor con el Arduino utilizamos la función digitalRead(pin). Esta función nos devuelve un valor que puede ser HIGH o LOW. Obtenemos el valor LOW cuando en ese pin no hay corriente eléctrica y el valor HIGH cuando sí la hay.

En nuestro caso se leerá el valor HIGH cuando el botón se presione, justo lo que necesitamos, ya que al presionar el botón la corriente eléctrica saldrá del pin de 5 volts, alimentando al pin 9.

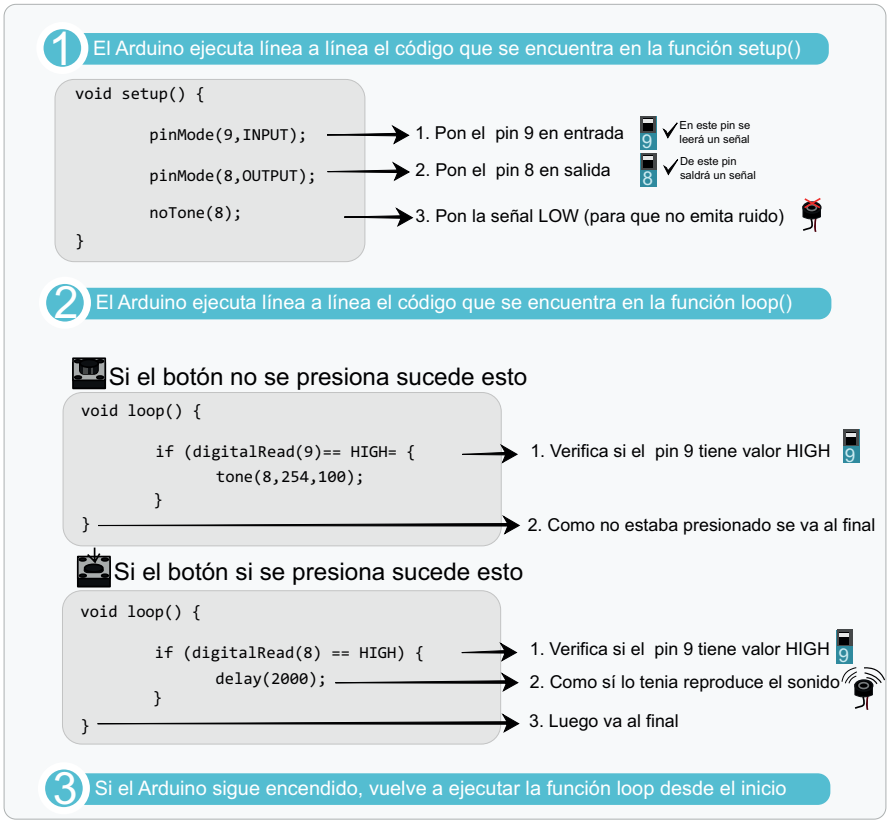
Código: Miniminipinano

```
void setup() {
  pinMode(8,OUTPUT); // le decimos a Arduino que usaremos el pin 8 como salida (para el del buzzer)
  pinMode(9,INPUT);// le decimos a Arduino que usaremos el pin 9 como entrada (para el botón)
  noTone(8); // nos aseguramos que no haya ningún sonido en el buzzer
}

void loop() {
  // vamos a leer el estado del botón usando la función digitalRead
  // y también usaremos una estructura de código llamada "if" o condicional
  if (digitalRead(9)==HIGH) { // Si en el pin 9 hay una señal HIGH es que se presionó el botón
    tone(8,254,100); // por lo tanto hay que reproducir el sonido por 100 milisegundos
  }
}
```

¡Cárgalo y pruébalo!

✓Si todo salió bien	El invento reproducirá un tono corto cada vez que presiones el botón.
×Si algo salió mal	No se escucha nada o el sonido se reproduce infinitamente.
Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado: comprobar que cada una de las patas del buzzer esté conectada, una al ground y otra al pin 8. ✓ Revisar especialmente el conexionado del botón y asegurarse que las patas que están siempre conectadas sean las del ground y las del pin 9 (mirar la parte trasera del botón y seguir el dibujo en el plástico negro).

Vamos a repasar qué es lo que sucedió cuando cargamos este código al Arduino:



Condicionales, if y bloques

La estructura if que usamos en el código anterior es la que nos permite ejecutar fragmentos de códigos o bloques dependiendo de si se cumple o no una condición.

La estructura básica de un if es la siguiente:

Fragmento de código

```
if (condicion) {
     instrucciones a ejecutar.
}
```

Las instrucciones que se encuentran dentro del if (las que están dentro de las llaves "{ }") solo se van a ejecutar si la condición se cumple, es decir si la condición que está dentro de los paréntesis tiene resultado verdadero.

Las condiciones son preguntas por valores de verdad. Por ejemplo, podemos preguntar si un número es mayor que otro, si es menor, si es igual o si es distinto, etc. Incluso podemos combinar condiciones usando operadores lógicos (AND y OR). Las siguientes son ejemplos de condiciones:

Expresión	Qué devuelve	Qué significa
8>0	verdadero	8 es mayor a 0
10<11	verdadero	10 es menor a 11
8>10	falso	8 es mayor a 10
10>=0	verdadero	10 es mayor o igual a 0
5==0	falso	5 es igual a 0
5!=0	verdadero	5 es distinto de 0
digitalRead(8) == LOW	Depende del valor en el pin 8	Si el resultado de la función read es LOW devolverá verdadero, sino falso

Los operadores lógicos son operaciones que se aplican sobre los valores de verdad así como –por ejemplo– la suma y la resta se aplican para los números.

Si quisiéramos ejecutar el código cuando se cumpla más de una condición podemos utilizar el operador AND, por ejemplo:

Fragmento de código

```
if ( (digitalRead(8)==HIGH) AND (digitalRead(10)==HIGH)) {
instrucciones a ejecutar.
}
```

En este caso el código se ejecuta solo si en el pin 8 se lee HIGH y en el pin 10 también se lee HIGH. Este ejemplo podría ser útil para prender un led solo cuando dos botones se presionan.

Otro operador útil es el OR, que siginifica “o”. Este operador devuelve verdadero si cualquiera de los dos valores es verdadero. Por ejemplo:

Fragmento de código

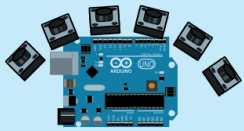
```
if ( (digitalRead(8)==HIGH) OR (digitalRead(10)==HIGH)) {
instrucciones a ejecutar.
}
```

En este caso el código se ejecutaría si se lee HIGH en el pin 8 o en el pin 10, o en ambos al mismo tiempo.

En otros proyectos veremos más ejemplos sobre expresiones para usar en los condicionales.

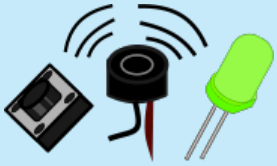
Ahora puedes seguir con el proyecto y sumar botones guiándote por los ejemplos anteriores, para hacer un minipiano de más teclas. Para ello, agrega más botones a los diferentes pines y elige los sonidos que quieras.

¡HORA DE EXPERIMENTOS Y EJERCICIOS!



1. ¡Agregar todos los botones que puedas!

2. Cambiar la duración de los tonos: haz que algunas notas duren más que otras.

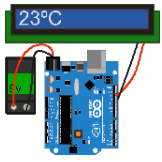


3. Agrega un led que se encienda cuando presiones cualquiera de los botones y se apague cuando no presiones ninguno. **Pista**: puedes usar los operadores lógicos AND y OR.

Termómetro portátil: mide, guarda, compara y muestra

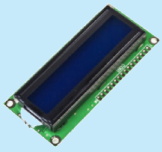
SOBRE ESTE PROYECTO

En este tercer proyecto usaremos una pantalla LCD para poder mostrar la temperatura de ambiente. Así podremos registrar la temperatura máxima, hacer sonar una alarma en caso de exceder cierta temperatura y lo que se nos ocurra. Además no necesitaremos estar conectados a la PC, ya que usaremos una batería de 9v para alimentar al Arduino.



Tags: analógico, condicionales, if, variables, pantalla LCD.

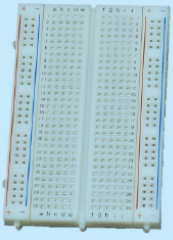
Elementos necesarios



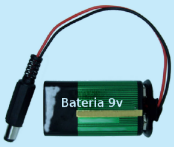
PANTALLA LCD
Esta pantalla nos permite mostrar dos líneas de 16 caracteres, comunicándose con el Arduino a través de los pines digitales.

POTENCIÓMETRO 10KΩ
Un potenciómetro es una resistencia a la que podemos controlar su valor ajustando una perilla.

SENSOR DE TEMPERATURA
Este componente hace que la energía eléctrica que lo atraviesa varíe según la temperatura a la que se encuentre.



PROTOBOARD
Es un lugar donde podemos conectar los cables y todos los componentes de nuestro proyecto de forma prolija y fácil.



BATERÍA 9V
La batería almacena energía que nos servirá para hacer que nuestros proyectos puedan ser portátiles.

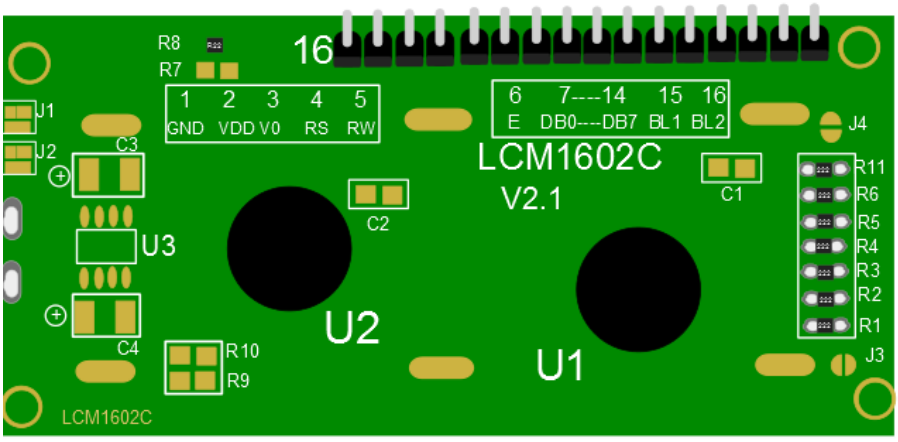
1 BOTÓN
Los botones simplemente unen los extremos de sus patas al ser presionados para permitir el paso de la corriente eléctrica.

MANOS A LA OBRA

Este proyecto incorpora muchos elementos y conceptos nuevos, pero nos va a abrir la puerta a más posibilidades en nuestros experimentos.

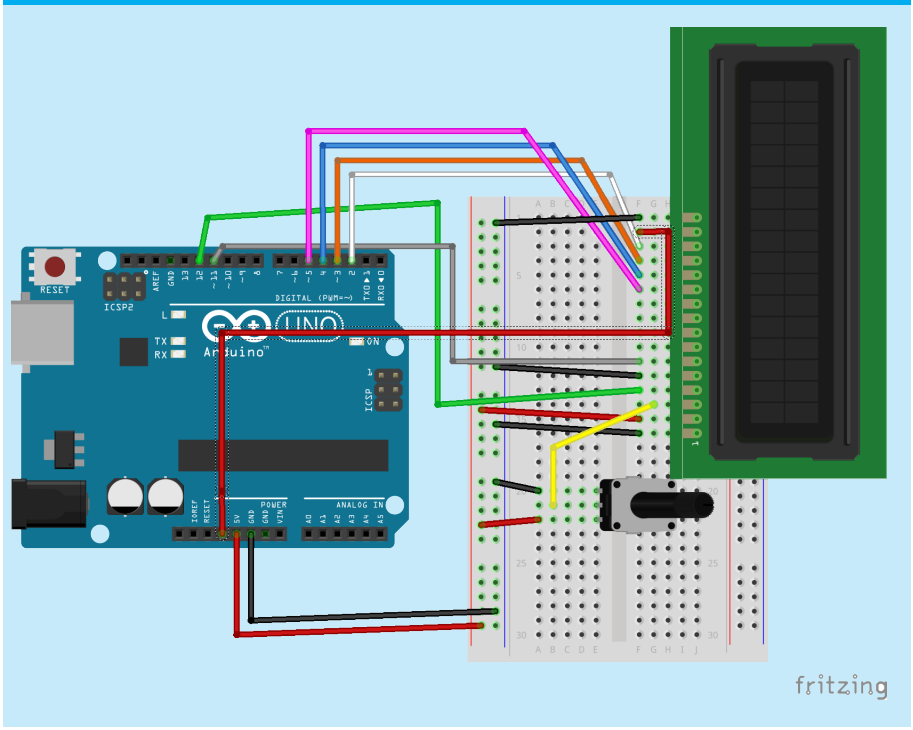
Comenzaremos con el más difícil de conectar hasta ahora por la cantidad de cables que lleva, la pantalla LCD. Nuestro Arduino se comunicará con esta pantalla a través de los pines digitales y esta nos dará la posibilidad de escribir en una pequeña pizarra donde entran ¡32 caracteres! Nada de televisores HD, acá nos conformamos con poco.

En la parte de atrás de la pantalla están numerados los pines de la misma, para saber cuál es la función de cada uno:



Hay que tener en cuenta que el cuadro que los identifica está al revés. El pin número 16 es el que está más a la izquierda y el 1 más a la derecha, en el extremo opuesto. Teniendo en cuenta esto hay que conectarlos al Arduino de la siguiente manera, utilizando el protoboard:

Esquema: conectando la pantalla LCD

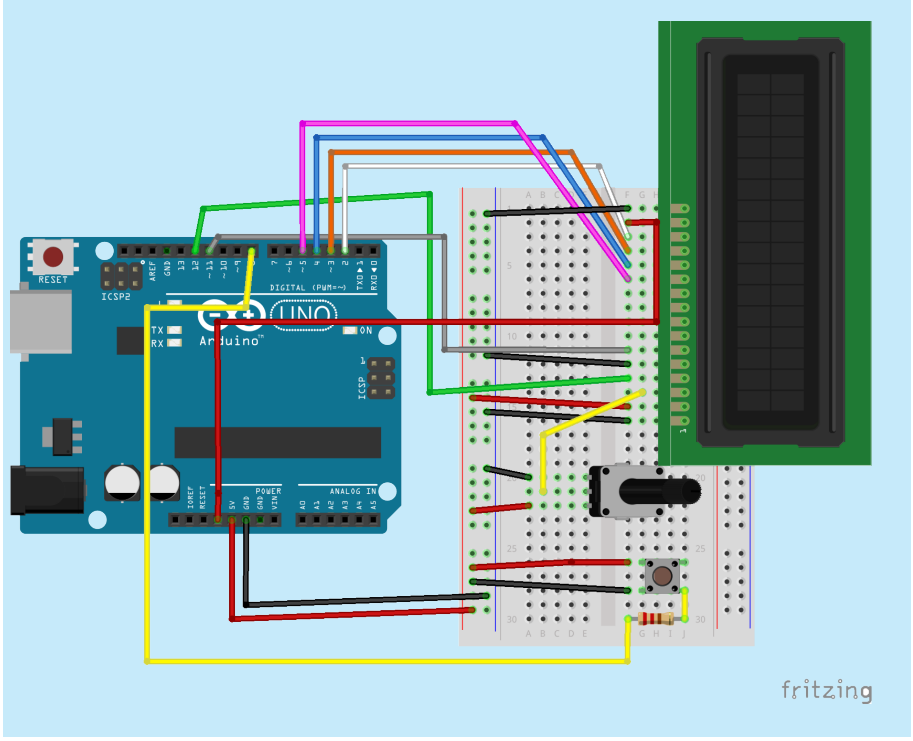


¡Es un verdadero lío! Acá va el paso a paso para simplificarlo:

a. Conecta la pantalla sobre el protoboard.
b. Haz que el pin 5V del Arduino se conecte a la alimentación (la línea roja) en el protoboard.
c. Haz que el pin GND del Arduino se conecte al del ground (la línea azul) en el protoboard.
d. El pin 1 de la pantalla va al ground (línea azul).
e. El pin 2 de la pantalla va al 5V del Arduino (línea roja).
f. El pin 3 de la pantalla va conectado a la pata del medio de un potenciómetro.
g. Las otras dos patas del potenciómetro van una a la línea de 5V y otra al ground.
h. El pin 4 de la pantalla va conectado al pin 12 del Arduino
i. El pin 5 de la pantalla va conectado al ground.
j. El pin 6 de la pantalla va conectado al pin 11 del Arduino.

Hasta ahí, esto debería verse más o menos así:

Esquema: conectando la pantalla LCD - Parte 1

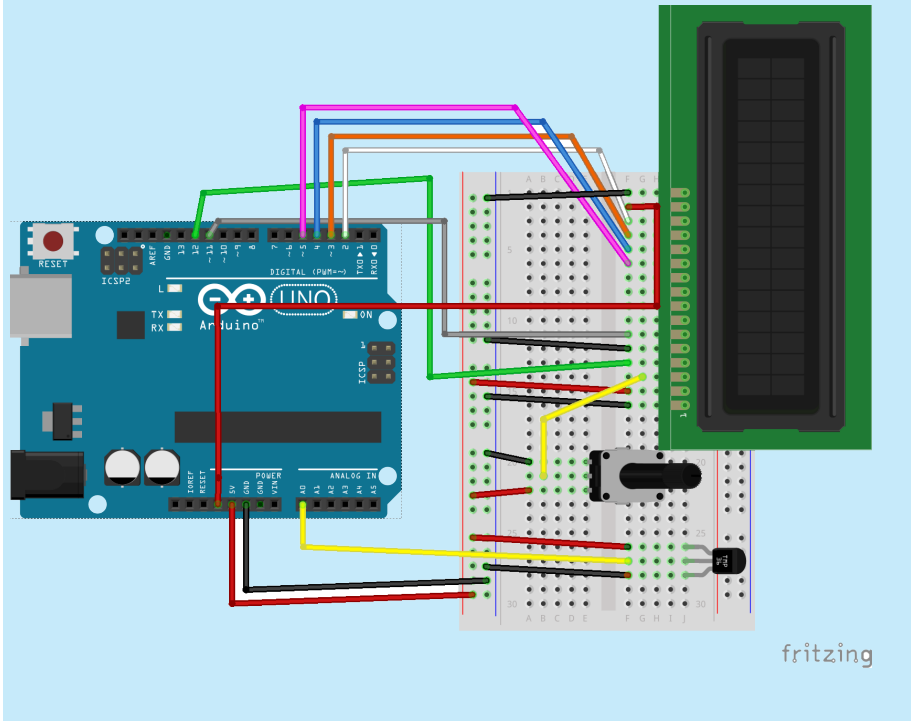


Luego vamos por el resto:

k. El pin 4 de la pantalla va conectado al pin 12 del Arduino.
l. El pin 5 va a la línea del ground.
m. El pin 7 de la pantalla va conectado al pin 2 del Arduino
n. El pin 6 de la pantalla va conectado al pin 3 del Arduino
o. El pin 5 de la pantalla va conectado al pin 4 del Arduino
p. El pin 4 de la pantalla va conectado al pin 5 del Arduino.

Debería verse así:

Esquema: conectando la pantalla LCD - Parte 2



Y por último, conecta los dos que faltaban:

q. El pin 16 de la pantalla va conectado al ground
r. El pin 15 de la pantalla va conectado al de 3.3V en el Arduino (no a la línea roja con los demás).

Si llegaste hasta acá, ¡ya terminaste de conectar la pantalla! Bien, ahora compila y carga el siguiente código al Arduino:

Código: Hola Hola pantalla LCD

```
#include <LiquidCrystal.h> // esta línea nos permite incluir código hecho por otras personas
//( o por nosotros mismos) y que se encuentra en otros archivos

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // se crea un objeto LCD con los pines del Arduino que
// están conectados a la pantalla

void setup() {
  lcd.begin(16, 2); // se le indica al objeto lcd cuántas columnas y cuántas filas tiene
// nuestra pantalla (16 caracteres en 2 líneas)
// Usamos nuestra pantalla para decir algo
lcd.print("Hola hola");
}

void loop() {
// nada por aquí
}
```

¡Cárgalo y pruébalo!

✓Si todo salió bien	El mensaje "Hola hola" se ve en pantalla.
×Si algo salió mal	La pantalla no se ilumina o no se ve el mensaje.
 Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos. ✓ Probar con el potenciómetro para ajustar el contraste (girándolo). ✓ Asegurarse de que el potenciómetro se haya conectado bien y no esté flojo.

Si todo anduvo bien ahora estás viendo el "Hola hola" en tu pantalla. Nota que el texto siempre se pone entre comillas cuando programamos.

Hay varias cosas nuevas también en el código. Primero está el "`#include <LiquidCrystal.h>`". Esta línea nos permite incluir código a un proyecto que ya estaba hecho. En este caso son un conjunto de funciones que nos permiten usar la pantalla LCD muy fácilmente.

Otra novedad es el crear un objeto con "`LiquidCrystal lcd(12, 11, 5, 4, 3, 2);`" pero esta explicación la dejaremos para más adelante. Lo importante es saber que para usar nuestra pantalla deberemos escribir "lcd.nombreDeFuncion". Y ahí aparece otra cosa nueva: las funciones. Por ahora diremos que las funciones son códigos escritos para resolver problemas específicos. Por ejemplo, el objeto lcd cuenta con las siguientes funciones:

Fragmento de código

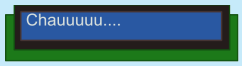
```
lcd.clear(); //borra toda la pantalla
lcd.setCursor(0,1);// mueve el lugar desde donde se escribe, en este ejemplo en la
línea de abajo
```

(Para ver más funciones de la librería LiquidCrystal, buscar en http://arduino.cc/en/Reference/LiquidCrystal)

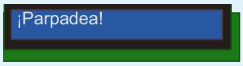
Más adelante aprenderemos a escribir nuestras propias funciones. Antes de seguir avanzando, ¿por qué mejor no hacemos algunos experimentos?

¡HORA DE EXPERIMENTOS Y EJERCICIOS!

Ahora que ya sabes un poco más sobre Arduino, por qué no intentás...

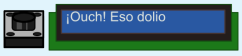


1. Hacer que el Arduino diga “Hola” y, después de un tiempo, “chau”.



2. ¡Hacer un mensaje titilante! **Pista**: prueba la función lcd.clear()

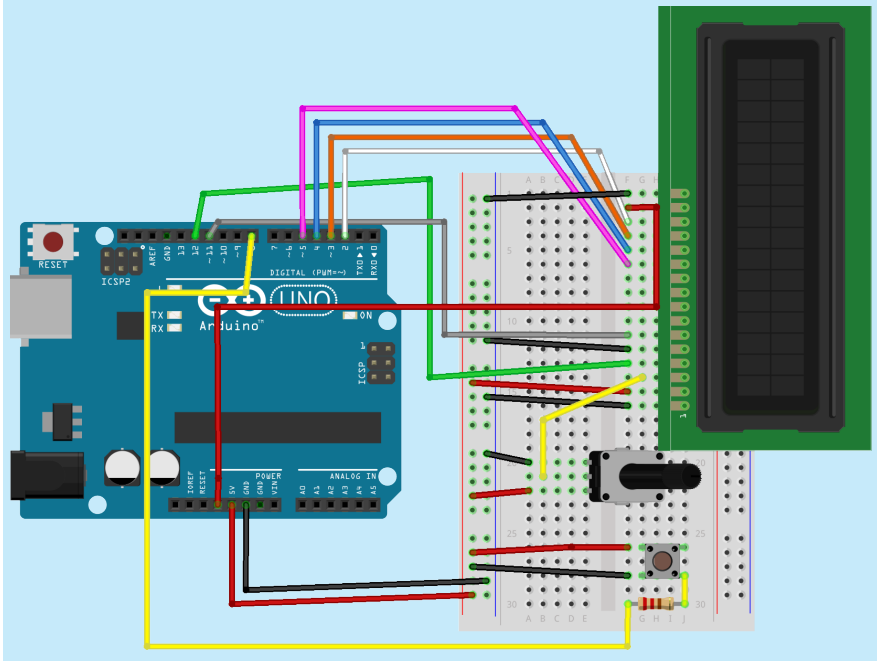
3. Escribir una palabra en el primer renglón y otra debajo. **Pista**: Usa la función lcd.setCursor()



4. Hacer que el Arduino diga “No me presiones” y que luego de presionar un botón diga “Ouch! Eso dolio”. **Pista**: usa los condicionales.

Para continuar es recomendable que completes el ejercicio 4 si aún no lo hiciste, ya que lo vamos a necesitar para nuestro experimento intermedio. ¿Alguna vez te ha pasado que has presionado un botón pero no sabes durante cuánto exactamente? ¿Esto te quita el sueño por las noches? Bueno, ya no te preocupes más porque nuestro experimento intermedio es un ¡contador de cuánto presionamos un botón! Para hacerlo, conecta un botón al pin 8 del Arduino.

Esquema: El contador inútil



Ya casi tenemos todo lo necesario para hacer nuestro invento millonario. Casi. Si queremos contar necesitamos un lugar donde guardar información, sino no sabremos cuánto tiempo lo hemos presionado.

Para ello necesitamos aprender sobre las variables. Las variables son lugares en la memoria de nuestra computadora donde podemos poner datos, leerlos y modificarlos. Imagínalo como si la computadora (o nuestro Arduino) tuviera dentro un montón de cajones donde podemos guardar y sacar cosas. La única restricción que tienen estos cajones es que solo podemos guardar una sola cosa y de un solo tipo, y que tiene que tener un nombre.

Por ejemplo, tenemos un cajón que se llama "Medias" donde guardo solamente una media, que puede ser roja, azul, o verde, pero no puede ser otra cosa que una media. No podría guardar allí, por ejemplo, un pañuelo.

En la computadora no podemos guardar una media pero sí números y letras. Estos son los tipos de datos que más usaremos.

Para crear una variable solo necesitamos saber de qué tipo va a ser y qué nombre le pondremos. Cualquier nombre que empiece con una letra es válido, siempre que no repita o tome el nombre de una función que ya exista.

Por ejemplo:

Fragmento de código

```
int miNumero = 0; // miNumero es una variable que contiene números, ahora el 0
int miOtroNumero = 3; // miOtroNumero es otra variable que contiene números, ahora el 3
string saludo = "Hola Hola"; // saludo es una variable que contiene texto
char letra ="A"; // letra es un carácter, es decir solo contiene una letra
```

La primera parte dice de qué tipo va a ser la variable: "int" para números enteros, "string" para palabras y texto, y "char" para letras. La segunda parte es el nombre de la variable y la ultima es donde le asignamos un valor. Veamos con lo que aprendimos qué cosas se pueden hacer y cuáles no.

Fragmento de código

```
int numeroLoco = 12; // esto si
int 123numeroMasLoco = 0; // esto no, el nombre de variable no empieza con una letra
int masNumero = "Algo"; // esto no, porque la variable la declaramos como de número entero
```

Para guardar cosas dentro de una variable usamos lo que se conoce como asignación, o el signo "=", los siguientes son ejemplos de asignaciones:

Fragmento de código

```
// primero declaramos las variables que vamos a usar con nombre y tipo
int a = 0;
int b = 0;
// luego las podemos usar
a = 10; //ahora a guarda el numero 10
b = 4; // ahora b guarda el numero 4
c = 5; // esto no se puede, no existe la variable c
```

También podemos usar el valor de otras variables para asignar. Por ejemplo:

Fragmento de código

```
// primero declaramos las variables que vamos a usar con nombre y tipo
int a = 0;
int b = 0;
int c = 0;
// luego las podemos usar
a = 4;
b = 2;
c = a + 1; // el valor de c es el valor de lo que tenga a (que es 4) + 1, es decir 5
c = a + b; // el valor de c será ahora el valor de lo que tenga a más el valor de lo
que tenga b
// es decir c será igual a 6
b = b * 2 // el valor de b será lo que valía b (que era 2) por 2, es decir 4
```

Bueno, basta de ejemplos, ahora sí pasemos a hacer nuestro flamante contador.

Código: Flamante contador

```
#include <LiquidCrystal.h> // Inlcuimos la libreria Liquid Crystal

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // Se crea un objeto LCD con los pines del Arduino

int contador= 0; //esta es nuestra variable, donde guardaremos el número de veces que
// lo presionamos

void setup() {
  pinMode(8,INPUT); // le decimos al Arduino que usaremos el pin 8 como entrada (el
// del botón)
  lcd.begin(16, 2); // se le indica al objeto lcd cuántas columnas y cuántas filas
// tiene nuestra pantalla
  lcd.print("Bienvenido al contador!");

  delay(1000);//mostramos el mensaje por 1 segundo
  lcd.clear(); //borramos todo lo que hay en la pantalla
}

void loop() {
  if (digitalRead(8)==HIGH) { // si el botón está siendo presionado
  contador= contador + 1; // incrementamos en 1 el valor del contador
  }
  lcd.setCursor(0,0); // ponemos el cursor al inicio de la pantalla
  lcd.print(contador);// escribimos en la pantalla el valor de la variable contador
}
```

¡Cárgalo y pruébalo!

✓Si todo salió bien	En la pantalla se ve la cantidad de tiempo que presionamos el botón.
×Si algo salió mal	La pantalla no se ilumina o no se ve el mensaje.
 Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos.

- ✓ Probar con el potenciómetro para ajustar el contraste (girándolo).
- ✓ Asegurarse de que el potenciómetro se haya conectado conectado y no esté flojo.
- ✓ Asegurarse de que el botón esté bien conectado.

Bien, pasemos a nuestro proyecto principal. Como vimos anteriormente, hasta ahora usamos los valores digitales HIGH y LOW para comunicarnos con los botones y leds, pero para medir la temperatura necesitamos usar más que solo dos valores: no podemos decir hoy hace HIGH grados de temperatura.

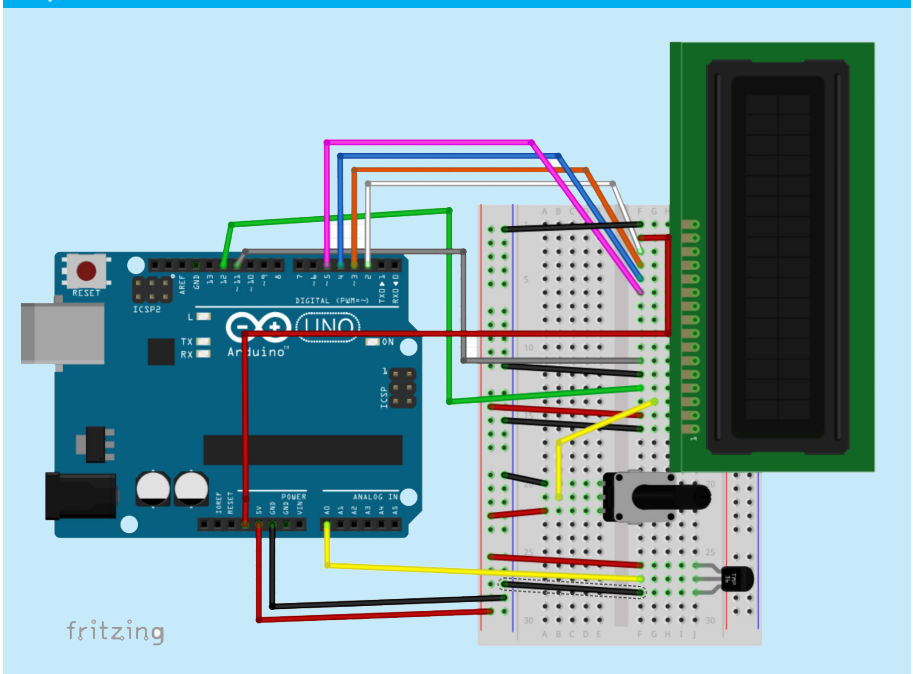
Las señales analógicas son la solución a nuestro problema. Estas señales, en lugar de tener solo dos valores posibles, tienen en cambio un amplio espectro. Las señales analógicas son continuas, lo que significa que tienen infinitos valores posibles. Sin embargo en nuestro Arduino las señales analógicas solo pueden ir del valor 0 al 1024.

El sensor que vamos a utilizar es, al igual que el potenciómetro, una resistencia variable que cambia su valor dependiendo de la temperatura que hay en el ambiente. Cuanto más calor hace, más energía pasa por el sensor; cuanto más frío hace, menos energía deja pasar.

La cantidad de electricidad que llegue a nuestro pin analógico en el Arduino va a determina el valor que leeremos. La menor cantidad corresponderá a 0 y la mayor a 1024.

Aquí va el esquema de conexión para nuestro sensor de temperaturas.

Esquema: El contador inútil

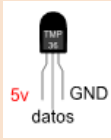


Para conectar el sensor TMP36 (está escrito en el sensor) hay que conectar sus patas externas, una al ground, otra al 5V y la pata del medio al pin analógico A0. Esos 6 pines de abajo que se llaman A0, A2, A3, A4 y A5 son los pines en los que podemos leer y escribir señales analógicas.

Por último, este es el código para nuestro flamante termómetro:

ATENCIÓN
Prestar especial atención a la conexión del sensor de temperatura ya que la pata que corresponde a GND y a 5V no debe intercambiarse. Si lo conecta al revés el sensor levantará temperaturas altas y puede quemar al tacto.

Modo correcto de conexión



Código: termómetro

```
#include <LiquidCrystal.h> // inlcuimos la libreria Liquid Crystal

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // se crea un objeto LCD con los pines del Arduino

int valorDelSensor = 0; // una variable de tipo número entero para guardar lo que leemos del sensor
float voltaje = 0.0; // una variable de tipo número flotante (con coma) para transformar esa medida
float temperatura = 0.0; // la variable que finalmente contendrá la temperatura

void setup() {
  pinMode(A0,INPUT); // decimos al Arduino que usaremos el pin 8 como entrada (el del botón)
  lcd.begin(16, 2); // se le indica al objeto lcd cuántas columnas y cuántas filas tiene
// nuestra pantalla
}

void loop() {
  valorDelSensor = analogRead(A0); // leemos el valor del sensor
  voltaje = (valorDelSensor/1024.0) * 5.0; // lo transformamos a voltaje
  temperatura = (voltaje - 0.5) * 100; // y esto a su vez lo transformamos a temperatura

  lcd.clear(); // limpiamos la pantalla
  lcd.print(temperatura);// escribimos en la pantalla el valor de la variable contador
  delay(500); // esperamos un poco antes de volver a tomar la temperatura
}
```

¡Cárgalo y pruébalo!

✓Si todo salió bien	En la pantalla se verá la temperatura.
×Si algo salió mal	La pantalla no se ilumina o no se ve el mensaje. La temperatura es incorrecta.
Qué revisar si no funcionó como debía...	✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos ✓ Probar con el potenciómetro para ajustar el contraste (girándolo).

- ✓ Asegurarse de que el potenciómetro se haya conectado bien conectado y no esté flojo.
- ✓ Asegurarse de que el sensor TMP36 esté bien conectado.
- ✓ Si la temperatura está mal fijarse si conectaste el sensor al ground y al pin de 5V del Arduino (y no al de 3.3V

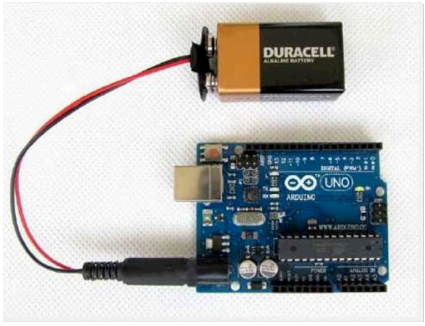
El sensor no nos da directamente la temperatura sino que nos da el voltaje que pasa por él (que dependerá de la temperatura) y ese valor a su vez hay que transformarlo en la temperatura. No es necesario que comprendas exactamente la fórmula, solo es importante que sepas cómo usar un sensor analógico y cómo medir la temperatura para tus proyectos.

Ahora sí, antes de terminar vamos a hacer que nuestro sensor muestre la temperatura máxima registrada usando más variables. Para eso vamos a crear una variable float (de numero con coma) llamada "temperaturaMax" y vamos a comparar el valor de la temperatura actual con esta. Si la temperatura actual es mayor a la temperaturaMax, entonces la temperatura actual es la máxima y la asignamos a la variable.

Código: termómetro

```
#include <LiquidCrystal.h> // incluimos la librería Liquid Crystal

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // se crea un objeto LCD con los pines del Arduino

int valorDelSensor = 0; // una variable de tipo número entero para guardar lo que leemos del sensor
float voltaje = 0.0; // una variable de tipo número flotante (con coma) para transformar esa medida
float temperatura = 0.0; // la variable que finalmente contendrá la temperatura
float temperaturaMax = 0.0;
void setup() {
  pinMode(A0,INPUT); // decimos al Arduino que usaremos el pin 8 como entrada (el del botón)
  lcd.begin(16, 2); // se le indica al objeto lcd cuántas columnas y cuántas filas tiene
// nuestra pantalla
}

void loop() {
  valorDelSensor = analogRead(A0); // leemos el valor del sensor
  voltaje = (valorDelSensor/1024.0) * 5.0; // lo transformamos a voltaje
  temperatura = (voltaje - 0.5) * 100; // y esto a su vez lo transformamos a temperatura
  lcd.clear(); // limpiamos la pantalla
  lcd.print("Temp:");
  lcd.print(temperatura);// escribimos en la pantalla el valor de la variable contador
  if (temperatura > temperaturaMax) { // si la temperatura actual es mayor a la temperatura máxima
    temperaturaMax = temperatura; // ahora la temperatura máxima es esa!
  }
  lcd.setCursor(0,1); // ponemos el cursor en la segunda línea
lcd.print("Max:");
  lcd.print(temperaturaMax);
delay(500); // esperamos un poco antes de volver a tomar la temperatura
  lcd.clear(); // limpiamos la pantalla
}
```

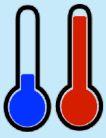
¡Bien! Ahora sí está terminado. Solo falta hacerlo portátil y eso es lo más fácil de todo. Simplemente desconecta al Arduino de la PC (es muy importante que nunca estén conectadas ambas cosas al mismo tiempo) y conecta la batería de 9V a la entrada de energía, como muestra la siguiente imagen.



¡HORA DE EXPERIMENTOS Y EJERCICIOS!

1. Hacer que registre temperatura mínima en lugar de máxima.

2. Agregar un botón que te permita resetear la temperatura máxima.

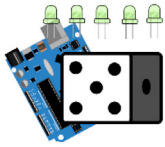


3. Hacer que el sensor solo muestre la temperatura máxima y mínima cuando presiones el botón, y que mientras el botón esté suelto muestre la temperatura actual.

El dado electrónico: números aleatorios y *loops*

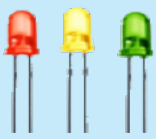
SOBRE ESTE PROYECTO

En este experimento crearemos un dado digital, agregando un sensor tilt a nuestro Arduino y usando números aleatorios.



Tags: random, aleatorio, tilt, loop, for.

Elementos necesarios



5 LEDS (ROJOS, AMARILLOS, Y VERDES)
Los leds son un tipo especial de diodos que emiten luz cuando la corriente eléctrica circula por ellos.

5 RESISTENCIAS 220W
Las resistencias son un componente electrónico que actúa como una barrera y deja pasar solo cierta cantidad de electricidad (medida en Ohmios).

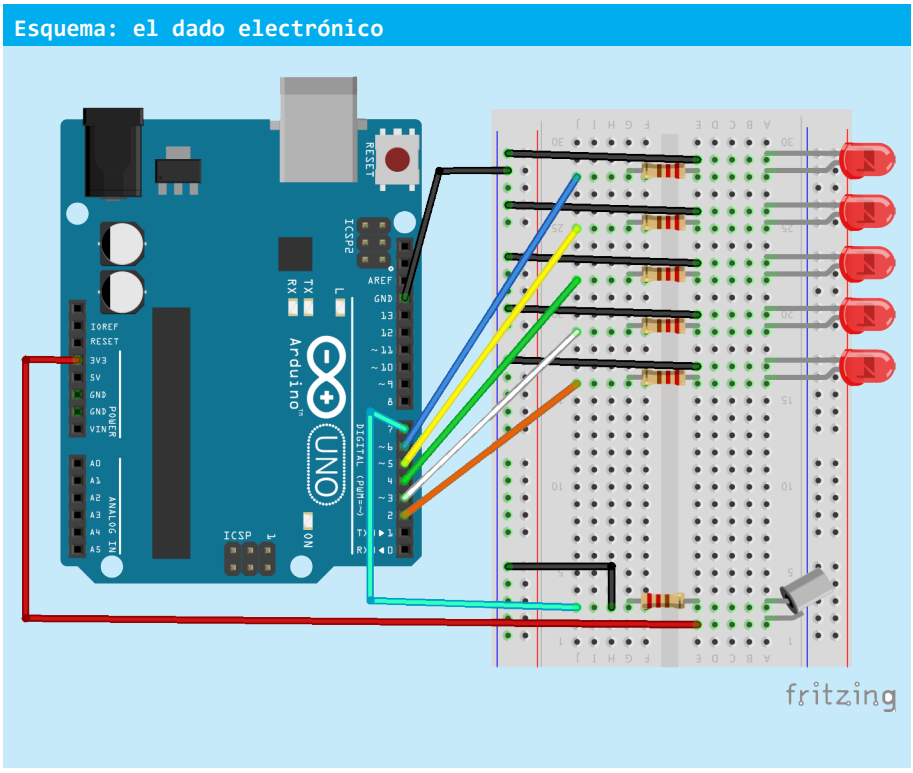
SENSOR TILT
Este sensor es un botón que se activa cuando una bolita que contiene dentro hace que los extremos de los pines se toquen.

MANOS A LA OBRA

En este proyecto bastante sencillo haremos uso de una nueva estructura de control que son los *loops* o bucles. Vamos primero a resolver este proyecto sin utilizar los bucles y luego veremos lo útiles que son.

Nuestro dado electrónico está compuesto en el hardware por 5 leds que van a indicar el numero que salió en el dado y un sensor llamado "tilt" que actúa como un botón. Cuando este se encuentra en posición vertical está activado y si lo inclinamos o damos vuelta se desactiva. Gracias a esto vamos a poder detectar cuando la persona que utiliza nuestro invento lo está sacudiendo en busca de suerte.

Esquema: el dado electrónico



Como vemos en el esquema, el sensor tilt se conecta igual que un botón. Lo que vamos a hacer es que cuando detectemos que no está haciendo contacto el sensor (significa que se está moviendo) generaremos un número aleatorio y mostraremos el resultado con la cantidad de leds que se prenden. Para ello usaremos el siguiente código:

Código: dado electrónico v1

```
int numero = 0;

void setup() {
  // ponemos los pines en modo salida para todos los leds
pinMode(1,OUTPUT);
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);

  // apagamos todos los leds
  digitalWrite(1,LOW);
  digitalWrite(2,LOW);
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
}


void loop() {
if (digitalRead(8)==LOW) {// si el dado electrónico está en movimiento

    numero=random(1,5);// si no generamos un número aleatorio de 1 a 5 (es decir puede
   // que salga 1,2,3,4, o 5;

    // apagamos todos los leds
    digitalWrite(1,LOW);
digitalWrite(2,LOW);
    digitalWrite(3,LOW);
    digitalWrite(4,LOW);
digitalWrite(5,LOW);
    // y prendemos solo los necesarios
    if (numero>=1) { // si el número es uno o más prendemos el primer led
      digitalWrite(1,HIGH);
    }
    if (numero>=2) { // si el número es dos o más prendemos el segundo led
      digitalWrite(2,HIGH);
    }
    if (numero>=3) { // si el número es tres o más prendemos el tercer led
      digitalWrite(3,HIGH);
    }
    if (numero>=4) { // si el número es cuatro o más prendemos el cuarto led
      digitalWrite(4,HIGH);
    }
    if (numero>=5) { // si el número es cinco o más prendemos el quinto led
      digitalWrite(5,HIGH);
    }
  }
}
```

¡Cárgalo y pruébalo!

✓Si todo salió bien	Al sacudir el sensor se modificaran aleatoriamente la cantidad de leds prendidos.
×Si algo salió mal	No se iluminara ningún led o siempre será la misma cantidad.

Qué revisar si no funcionó como debía...

- ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB.
- ✓ Revisar el conexionado varias veces, en especial el del sensor tilt.
- ✓ Asegurarse de que el sensor tilt se mueva realmente (se escuchará la bolita de metal moviéndose dentro).

Bueno, hasta acá todo bien, ya que solo tenemos 5 leds para prender... ¿pero qué sucede si en lugar de tener 5 tuviéramos 100 leds? ¿Tendríamos que agregar líneas y líneas y líneas de código? Claro que no, para eso están los queridos bucles. Un bucle es una estructura de código que nos permite hacer que algo se ejecute una y otra vez mientras se cumpla una condición. Vamos a ver con un ejemplo cómo hacemos para apagar todos los leds usando el bucle llamado "for".

Fragmento de código

```
1  for (int pin = 0; pin <=5; pin++) {
2  digitalWrite(pin,LOW);
3  }
```

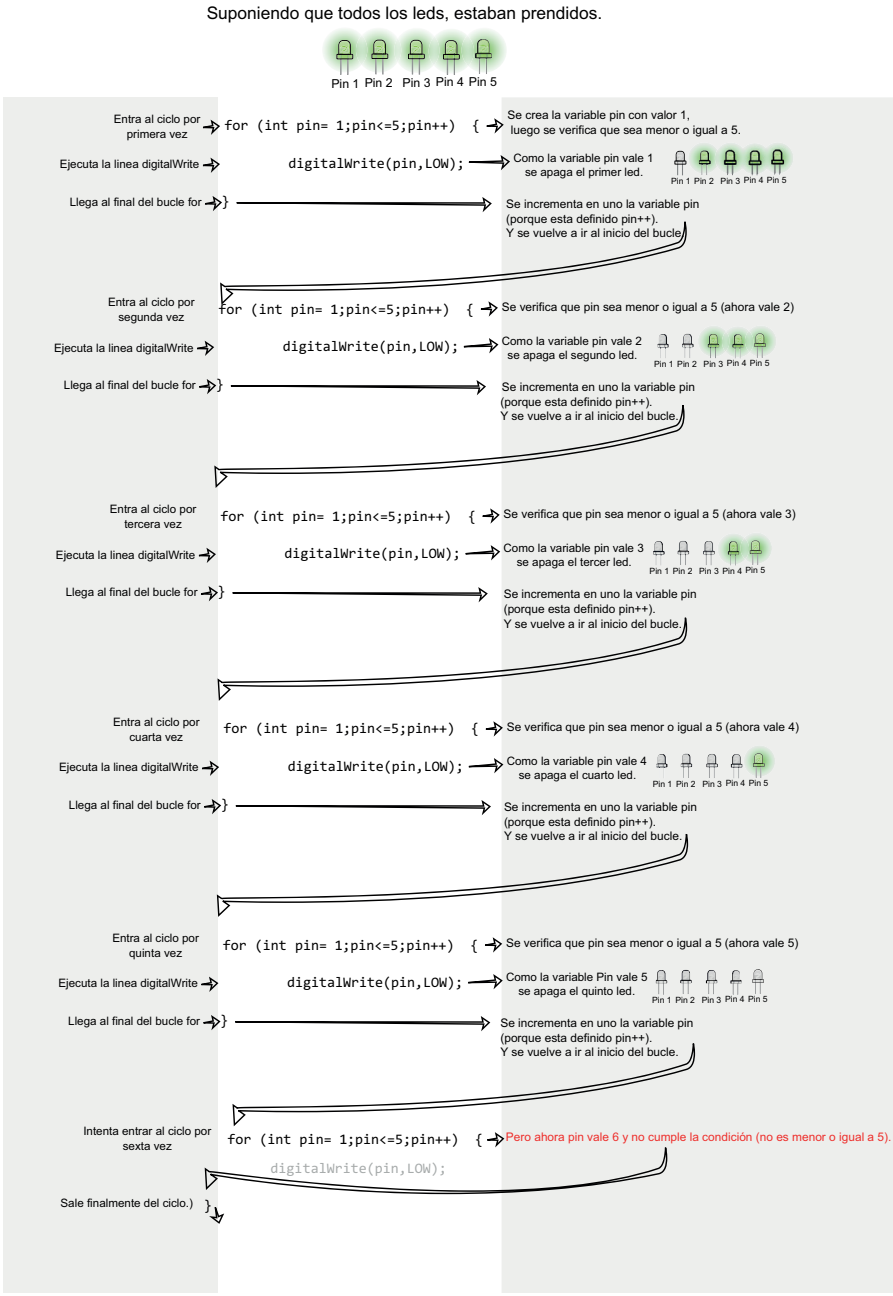
Un for está formado por un paréntesis que contiene tres declaraciones importantes separadas por ";" (punto y coma).

En la primea parte "`int pin = 1`" se crea una variable que se conoce como iterador. Es la variable que usaremos en el for para contar la cantidad de veces que repetimos algo. En nuestro caso vamos a ponerle 1 de valor inicial.

La segunda parte dice "`pin <=5`". Lo que pongamos dentro del for se va a repetir hasta que deje de cumplirse esta condición. En este caso se va a dejar de repetir cuando la variable "pin" valga más que 5.

La tercera y última parte, dice "`pin++`" lo que significa que luego de ejecutar el código a la variable "pin" se le sumará 1.

¿Un poco confuso, no? Vamos a ver un paso a paso de cómo ejecutaría esto Arduino:



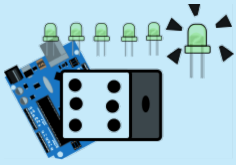
En resumen, el ciclo for se va a repetir incrementando el valor de su contador (en el ejemplo, la variable pin) hasta que no se cumpla más su condición. Mientras se cumpla esta condición el programa quedará atrapado dentro de éstas líneas ejecutando una y otra vez las mismas instrucciones (e incrementando su contador, claro está).

Ahora sí, vamos a mejorar nuestro código utilizando los bucles for.

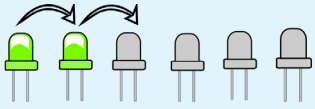
Código: dado electrónico v2

```
int numero = 0;
void setup() {
  for (int pin=1;pin<=5;pin++) {// desde que el pin valga 2 hasta que valga más que 6
    pinMode(pin,LOW);// ponemos a cada pin en modo salida
    digitalWrite(pin,LOW);// y lo apagamos
  }
  pinMode(8,INPUT); // el pin del sensor tilt
}
void loop() {
    if (digitalRead(8)==LOW){ // si el pin del sensor tilt está en low quiere decir que
    // está en movimiento
    // apagamos todos los leds
    for (int pin=1;pin<=5;pin++) {// desde que el pin valga 2 hasta que valga más que 6
      digitalWrite(pin,LOW);// lo apagamos
    }
    numero=random(1,6);// generamos un número aleatorio que va de 1 a 5
    for (int pin=1;pin<=5;pin++) {// por cada pin
      if (pin<=numero) {// verificamos si el pin es menor igual al número generado
    // aleatoriamente
        digitalWrite(pin,HIGH);// si es así prendemos el led
      }
    }

  }
}
```

¡HORA DE EXPERIMENTOS Y EJERCICIOS!



1. Agregar un led más para que sean 6, al igual que un dado.

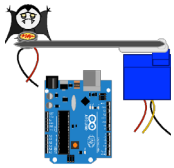


2. Hacer que al iniciar los leds se prendan de a uno hasta encenderse todos, usando un for, y luego se apaguen.

¡A salvar al vampiro!: servomotores, sensores de luz y potenciómetros

SOBRE ESTE PROYECTO

En este quinto proyecto utilizaremos un sensor de luz (una fotorresistencia) para simular a un pequeño vampiro que se moverá utilizando un servomotor. La idea del juego es intentar salvarlo hasta que se acabe el tiempo.



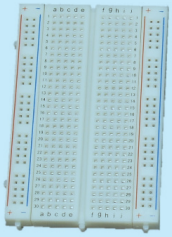
Tags: servomotores, fotorresistencias, funciones, condicionales, analógico, monitor serial

Elementos necesarios

FOTORRESISTENCIA
La fotorresistencia, como su nombre lo indica, es una resistencia cuyo valor varía dependiendo de cuánta luz recibe.

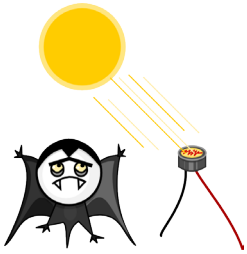
BUZZER
Los buzzers son capaces de emitir sonidos porque en su interior cuentan con un electroimán que hace vibrar una placa de metal.

SERVOMOTOR
Un servomotor es un motor de movimiento controlado que tiene un sensor para saber cuántos grados gira y en qué posición está.



PROTOBOARD
Es simplemente un lugar donde podemos conectar los cables y todos los componentes de nuestro proyecto de forma prolija y fácil.

1 RESISTENCIA 1KΩ



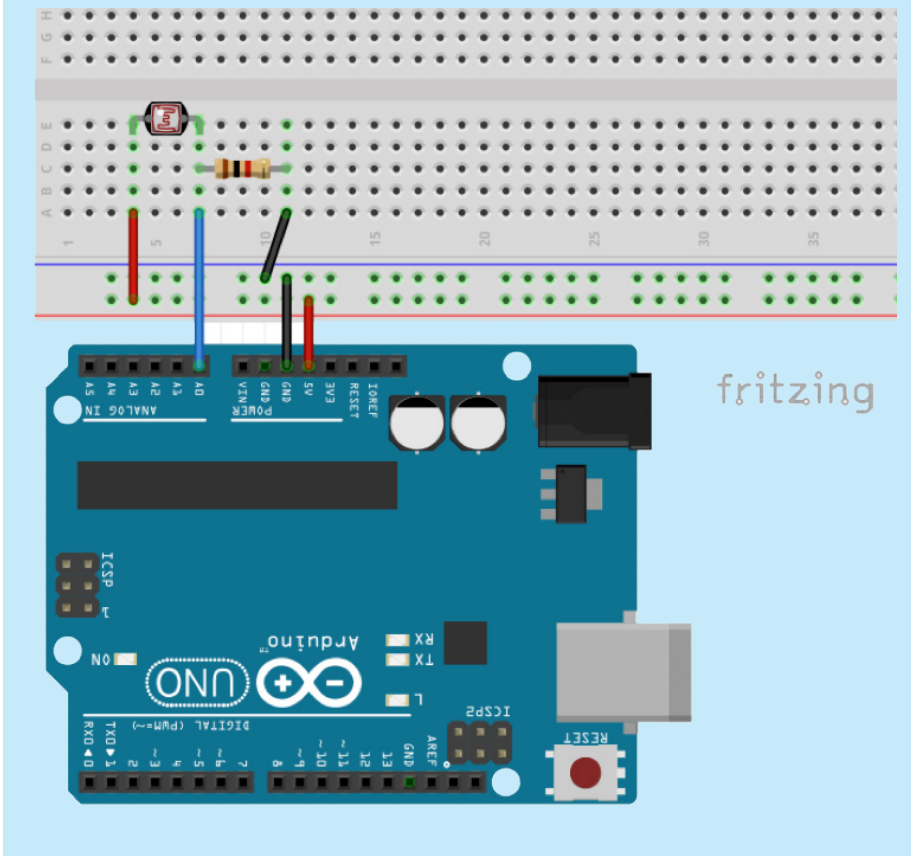
En este proyecto vamos a hacer un juego muy sencillo y benévolo en el que debemos salvar la vida de un vampiro. Empezaremos para ello por simular nuestro vampiro electrónicamente con una fotorresistencia.

Una fotorresistencia está compuesta por un material cuya resistencia al paso de la corriente varía según la cantidad de luz que recibe. Cuanta más luz haya menor será la resistencia que opone.

De este manera nosotros vamos a ser capaces de detectar cuánta luz hay en el ambiente, lo que nos abre la puerta a nuevos proyectos. Por ejemplo: podríamos hacer una luz que se prenda solo de noche, o podríamos crear un aparato para medir la intensidad de luz que recibe una planta para saber donde nos conviene colocarla o bien... ¡armar un vampiro electrónico!

El siguiente esquema te muestra cómo conectarla al Arduino. Utilizaremos el pin analógico A0, la salida de 5v y una resistencia de 1kΩ

Esquema: creando el vampiro



Empezaremos a experimentar con el siguiente código:

Código: vampiro01

```
void setup() {
pinMode(A0,INPUT);// definimos que usaremos al pin A0 para recibir valores
Serial.begin(9600);// creamos una conexión serial por defecto
}

void loop() {
Serial.println("Leyendo" + analogRead(A0));// enviamos el valor de A0 usando la
// conexión serial)
delay(100); // esperamos un momentito para no llenar el monitor serial con tantos datos
}
```

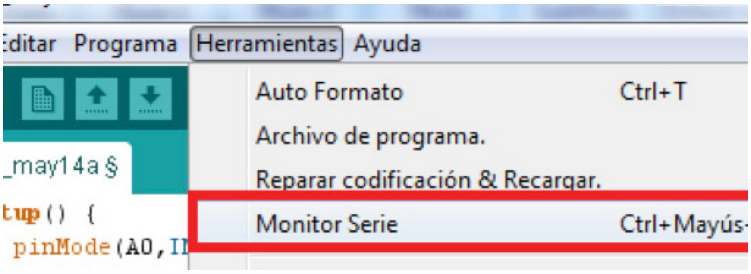
Antes de cargarlo y probarlo, vamos a hablar sobre la conexión serial. La siguiente línea crea una conexión serial (un protocolo de comunicación), la cual podemos usar para enviar y recibir datos.

Fragmento

```
Serial.begin(9600);//creamos una conexion serial por defecto
}
```

Cuando tenemos conectado el Arduino a la computadora mediante el cable USB tenemos la posibilidad de usar esta conexión no sólo para cargar los programas que hacemos, sino también para enviar y recibir información.

Una forma fácil de ver qué información mandamos desde el Arduino es utilizar el "monitor serial" que incluye nuestro IDE Arduino (Herramientas -> Monitor Serie).



Ahora sí, cárgalo y pruébalo. Luego abre el monitor serial para ver lo que envía nuestro Arduino.

✓Si todo salió bien	En el monitor serial aparecerán las lecturas del sensor.
×Si algo salió mal	No vas a ver nada en el monitor serial o verás caracteres raros.

Qué revisar si no funcionó como debía...

- ✓ Fíjate que el puerto del monitor serial sea el correcto (debe ser el mismo que se usa cuando cargas los programas).
- ✓ Si se ven caracteres “raros” asegúrate que la conexión del monitor serial sea también 9600 (esa es la cantidad de bits que envía por segundo).
- ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB.
- ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos

Prueba tapar el sensor, acercarlo a una fuente de luz o también ponerle encima una linterna o la pantalla de un celular, para ver cómo se modifican esos valores.

Bien, ya tenemos la piel sensible del vampiro. Ahora vamos a simular la vida de nuestro personaje usando una variable que se llame “salud”.

Código: Vampiro02

```
int salud = 1000;
int luzRecibida = 0;

void setup() {
pinMode(A0,INPUT);//definimos que usaremos al pin A0 para recibir valores
Serial.begin(9600);//creamos una conexión serial por defecto
}

void loop() {
   if (salud>0) { //si nuestro vampiro tiene salud chequeamos cuánta luz le llega
       luzRecibida = analogRead(A0);
Serial.println("Leyendo" + luzRecibida);// vemos cuánta luz recibe)
delay(100); // esperamos un momentito para no llenar el monitor serial con tantos datos
       if (luzRecibida> 400) { // si recibe más luz que cierto valor que nosotros establecemos
           salud -=1; // le restamos salud a nuestro vampiro
       }
   } else { // sino tiene salud
       Serial.println("He muerto :("); // imprimir el mensaje de "he muerto"
       delay(100); // esperamos un momentito para no llenar nuestro monitor serial.
   }

}
```

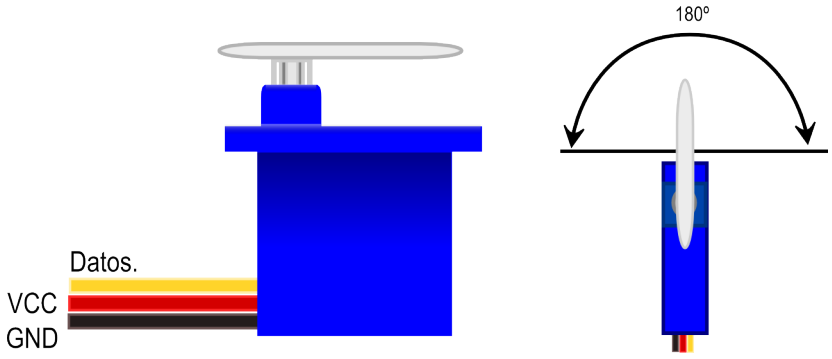
¡Cárgalo y pruébalo!

✓Si todo salió bien	En el monitor serial aparecerán las lecturas del sensor. Cuando el vampiro haya muerto, debería verse siempre el mismo mensaje de "He muerto :("
×Si algo salió mal	El vampiro muere demasiado rápido. No se ven los mensajes. El vampiro no muere.

Qué revisar si no funcionó como debía...

- ✓ Comprueba que los valores leídos se correspondan con el valor de 400 que establecimos. Puede que los valores de luz siempre sean menores o mayores y que debas modificar este valor.
- ✓ Fíjate que el puerto del monitor serial sea el correcto (debe ser el mismo que se usa cuando cargas los programas).
- ✓ Si se ven caracteres “raros” asegúrate de que la conexión del monitor serial sea también 9600 (esa es la cantidad de bits que envía por segundo).
- ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB.
- ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos.

Ahora falta agregarle movimiento al vampiro y utilizaremos para eso los servos. Un servo es un motor que tiene un movimiento limitado de 180° pero que nos permite moverlo con gran precisión y sabiendo exactamente en qué ángulo se encuentra todo el tiempo.



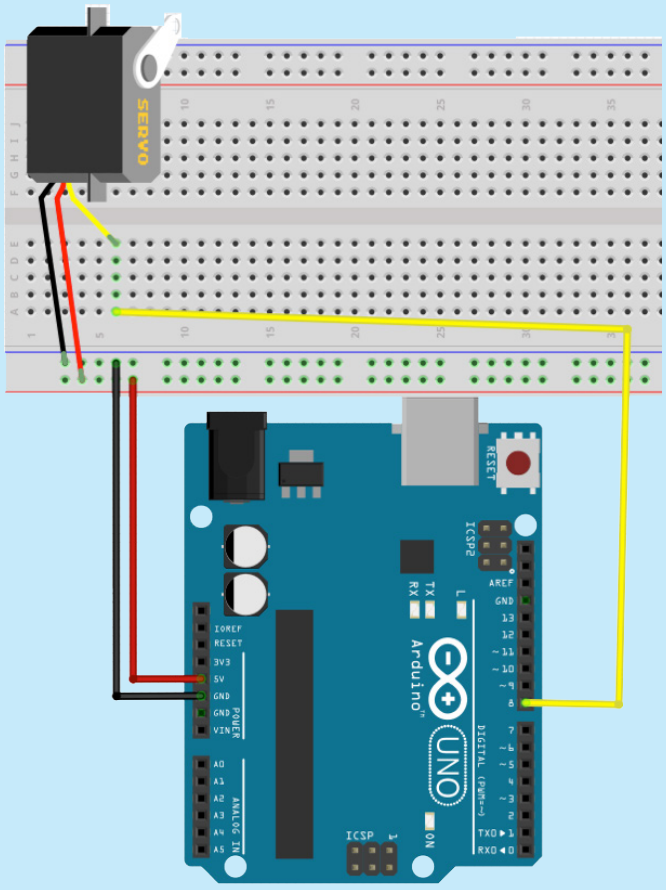
Servomotor con movimiento limitado de 180°

Para conectarlo a nuestro Arduino hay que tener en cuenta que utiliza tres cables, dos de ellos para la alimentación y uno de ellos para darle las ordenes y leer el ángulo en el que se encuentra.

Desde el punto de vista de la programación utilizamos lo que se conoce como librería. Una librería es código escrito por otra persona (o varias) que agrega funciones adicionales a nuestro repertorio y que nos facilitará la creación de los programas de Arduino. En esta caso utilizaremos la librería Servo.

Haz el siguiente esquema con tu Arduino y carga el código:

Esquema: Jugando con el servomotor



Código: JugandoConServo01

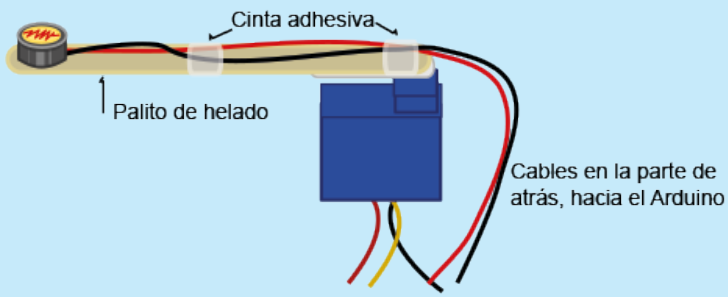
```
#include <Servo.h> //incluimos la libreria Servo

Servo myservo;  // creamos un objeto que manejará al servo


void setup() {
  myservo.attach(8);// le decimos al servo qué pin usará
  myservo.write(90); // escribimos la posición 90° para que el servo apunte a su ángulo medio
  delay(1000);  // esperamos 1 segundo
}

void loop() {
      myservo.write(0);// ponemos en 0° ángulo
      delay(1000); // esperamos 1 segundo
      myservo.write(180); // ponemos en 180° el ángulo
      delay(1000);// esperamos 1 segundo
}
```

✓Si todo salió bien	El servo apunta primero a una posición central. Luego de un extremo al otro describiendo 180°.
×Si algo salió mal	El servo no se mueve.
Qué revisar si no funcionó como debía...	✓ Que el servo, con su cable amarillo, esté conectado al pin 8 ✓ Que el servo reciba energía de 5V y que tenga conectado el GND. ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos.

Como pudiste ver, es muy sencillo manejar al servo utilizando la librería. Usamos la función attach para asignarle un pin para manejar el servo, y luego la función write, para poder setear el ángulo que queremos que tenga.

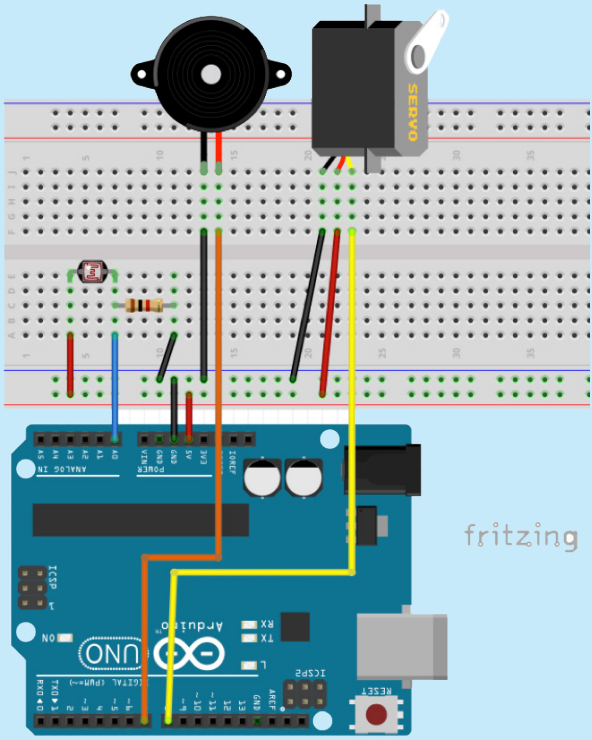
Vamos ahora a juntar ambas cosas y a poner la fotorresistencia sobre el vampiro. La idea es usar un palito de helado, o cualquier otro palito, y adherirlo al servomotor. Luego, en la punta de este, colocar la fotorresistencia con cuidado para que los cables no se enreden con el movimiento. Puedes usar el siguiente esquema:

Esquema: armado del vampiro03



Ahora que la fotorresistencia está en el extremo del palito de helado, y el palito de helado está adherido (con cinta adhesiva o un tornillo) al soporte de plástico del servo, podemos arrancar con el conexionado. Lo que vamos a hacer es conectar el servo y la fotorresistencia al mismo tiempo. Además, vamos a agregar un buzzer para comunicarnos con el jugador.

Esquema: Jugando con el servomotor



Código: JugandoConServo01

```
#include <Servo.h> //incluimos la librería Servo

Servo myservo;  // creamos un objeto que manejará al servo
int luz = 0; // creamos una variable para indicar la luz que detecta el sensor
int salud = 10000; // tiene 30 puntos de vida
int luzQueMata = 850; // máxima luz que puede tolerar el vampiro
int tiempoEnEsteLugar = 0;// cuenta hace cuánto que no se cambia el movimiento del servo
int tiempoDeEspera = 5000; // el tiempo de espera máximo para un movimiento
int puntos = 0;
bool sono= false;


void setup() {
  pinMode(A0,INPUT); // usaremos el pin A0 en modo input
  pinMode(7,OUTPUT);// usaremos el pin 7 para el buzzer
  myservo.attach(8);// le decimos al servo qué pin usará
  myservo.write(90); // escribimos la posicion 90° para que el servo apunte a su ángulo medio
  // música de inicio del juego
  tone(7,440);
delay(300);
  tone(7,540);
  delay(300);
  tone(7,640);
  delay(200);
tone(7,540);
  delay(400);
  noTone(7);
  delay(300);
}
```

```
void vampiroVivo() {// esta es una función que creamos nosotros
    if (tiempoEnEsteLugar > tiempoDeEspera) {// si ha estado en este lugar mucho tiempo
       myservo.write(random(0,180)); // hace que el servo se mueva a un ángulo aleatorio
       tiempoEnEsteLugar = 0; // reinicia el contador de tiempo
    } else {// en caso contrario, si todavía no pasó el tiempo de espera
      tiempoEnEsteLugar+=1;// acumula tiempo
    }

    if (luz > luzQueMata-50) { // si hay más luz que la que puede soportar el vampiro
   // (más un margen de error)
      salud-=1;// restamos salud
      tone(7,50); // reproducimos un sonido para que el jugador sepa
    } else {
      noTone(7); // si no se le quita salud no reproducimos ningún sonido
    }
}

void vampiroMuerto() {// esta es otra función que nosostros creamos
  if (!sono) { // si no hizo la música
tone(7,220);
    delay(400);
    tone(7,200);
    delay(400);
    tone(7,180);
delay(400);
    tone(7,100);
    delay(800);
    noTone(7);
    sono = true; // ahora sí la hizo
  }
}

void loop() {
  luz = analogRead(A0); // leemos el pin A0 y almacenamos el valor en luz

  if (salud>0) { // si el vampiro está vivo
    vampiroVivo(); // ejecute la función vampiroVivo
  } else {
    vampiroMuerto(); // ejecute la función vampiroMuerto
  }

}
```

✓Si todo salió bien	El servo apunta primero a una posición central y se escucha la música de inicio. Acto seguido, el servo se mueve de forma aleatoria haciendo pequeñas pausas. Cuando no hay sombra sobre el sensor se emite un sonido. Si pasa suficiente tiempo destapado se escuchara la música de fin del juego y no se moverá más.
×Si algo salió mal	No se presenta ninguno o solo alguno de los comportamientos deseados.
 Qué revisar si no funcionó como debía...	✓ Que todos los elementos estén conectados a los pines correspondientes. Cuantos más elementos, más lugar a confusión. ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos.

Es un código largo, sí, pero muy simple. La novedad que llama la atención en

este código es que podemos crear funciones en nuestro código. Las funciones vampiroVivo y vampiroMuerto, fueron creadas por nosotros. Vamos a definir más formalmente el concepto de este nuevo elemento del que disponemos para crear nuestros programas.

Una función es un código de programa que tiene un nombre asociado y puede ser utilizado en cualquier parte del programa invocándolo a través de un denominador. Una función también puede recibir parámetros y devolver valores, y nos resulta muy útil para organizar el código y para reutilizarlo.

La función tone nos permite emitir un sonido en un buzzer. Recibe distintos parámetros de entrada como el pin, la frecuencia y el tiempo y no devuelve ninguno.

La función analogRead por otro lado, recibe un parámetro, el número del pin, y retorna un valor que corresponde a la lectura de ese pin.

Para definir una función se hace de la siguiente manera:

Fragmento de código

```
tipoDeDatoQueDevuelve nombreDeFuncion(parametro1,parametro2, etc) { // llave que abre
     // código de la función
} // llave que cierra
```

Cuando nuestra función va a retornar un valor, tenemos que especificar el tipo de este valor. Por ejemplo, en la siguiente función recibe dos números como parámetro, los multiplica y devuelve el producto.

Fragmento de código

```
int multiplica(num1,num2) { // la función multiplicar
int res;
    res = num1 * num2;
return resultado;
} // llave que cierra

int pesoPaquete = 2;
int cantidadPaquetes = 10;
int pesoTotal = 0;

pesoTotal = multiplica(pesoPaquete, cantidadPaquetes);
```

Para retornar un valor de una función, se utiliza la palabra clave “return”, y en general se coloca al final de la función. Cuando se ejecuta la línea return, se devuelve el valor y se termina la ejecución de la función.

Para usar la función simplemente se la llama por su nombre, “multiplicar”, se eligen los parámetros pesoPaquete y cantidadPaquetes, y el valor que devuelve se lo asigna a una variable: “pesoTotal”.

En el siguiente y último proyecto haremos un mayor uso de las funciones, pero por ahora tenemos más cosas para hacer...

¡HORA DE EXPERIMENTOS Y EJERCICIOS!

Ahora que ya sabes un poco más sobre Arduino por qué no intentás...

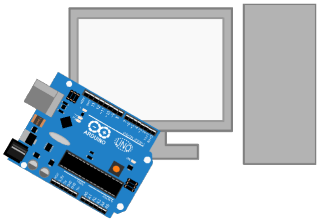
1. Que el valor que toma el sensor sea automático. Es decir que se determine antes de comenzar el juego

2. Conectar la pantalla LCD para que sume un puntaje y lo muestre.

3. Guardar el puntaje más alto de la partida y hacer que se muestre cuando finalice el juego. En caso de que un jugador supere el *highscore* (récord de puntos), debería ser felicitado y sonar la misma música que al inicio del juego. ¿Se te ocurre cómo hacerlo usando funciones? También puedes agregar un botón para reiniciar la partida.

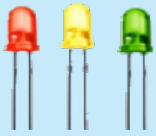
Mensajes: la comunicación entre la computadora y Arduino

SOBRE ESTE PROYECTO



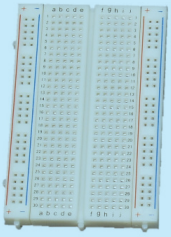
En este último proyecto haremos que nuestra PC y el Arduino se comuniquen a través del puerto serial. Para ello haremos uso de otro lenguaje de programación y crearemos dos programas: uno para el Arduino, como siempre, y otro para nuestra computadora usando Processing.

Elementos necesarios



LEDS
Los leds son un tipo especial de diodos que emiten luz cuando la corriente eléctrica circula por ellos.

POTENCIÓMETRO 10KΩ
Un potenciómetro es una resistencia a la que podemos controlar su valor ajustando una perilla.



PROTOBOARD
Es simplemente un lugar donde podemos conectar los cables y todos los componentes de nuestro proyecto de forma prolija y fácil.

MANOS A LA OBRA

Es posible que en nuestros proyectos nos encontremos con la necesidad de contar con funciones que tenemos disponibles en nuestra computadora o bien que necesitemos un poder de cómputo mayor al que ofrece Arduino. Por eso, ¡es hora de conectar ambos mundos! De esta manera obtendremos, por un lado, la flexibilidad de Arduino con todos sus sensores y actuadores, y por otro, la potencia y las herramientas que posee la PC.

Para poder hacerlo posible vamos a tener que instalar en nuestra computadora un nuevo IDE para programar en Processing. Es muy similar al lenguaje que usamos con Arduino pero que nos permitirá crear programas para ejecutarlos en nuestra computadora. Además vamos a necesitar usar el puerto serial, que ya habíamos utilizados antes, para poder mandar y recibir los mensajes entre la PC y Arduino.

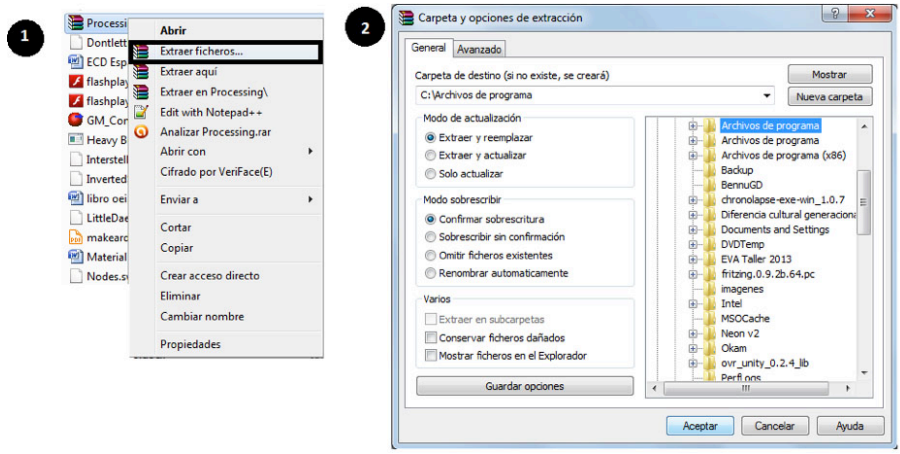
Vamos a empezar por instalar Processing. Para ello:

1) Descarga el entorno Processing

Ingresa en la página de Processing y elige si quieres o no hacer un donativo. Luego elige el sistema operativo en el cual lo quieres instalar.

2) Descomprime el entorno Processing

Busca el archivo en las descargas, haz clic derecho y extráelo en la carpeta que gustes. Un buen lugar puede ser C:/Archivos de Programa/Processing.



Ahora que tenemos instalado Processing vamos a ver un poco cómo funciona con un pequeño ejemplo de programa hola mundo. Para ejecutarlo, busca en la carpeta que descomprimiste el archivo processing.exe.

Código Processing: HolaMundo

```
void setup() {// es igual a la función setup de Arduino
  size(640,480); // esta línea es obligatoria y debe ser la primera
  // nos permite definir el tamaño de la ventana
  background(255,255,255);   // dibujamos un fondo blanco
  frameRate(60); // establecemos la frecuencia con que se grafica en pantalla
  noStroke(); // no usaremos borde en las figuras que dibujamos
}

void draw() { // es equivalente a la función loop en Arduino, solo que nos
// permite dibujar en pantalla.
  fill(0,0,0); // le indicamos que vamos a dibujar una figura con
  // color de relleno negro
  ellipse(mouseX,mouseY,10,10); // dibujamos una elipse en la posición del mouse
  text("HOLA MUNDO",width /2, height/2); // escribimos "hola mundo" en la pantalla
}
```

✓Si todo salió bien	Se creará una ventana, dibujara "HOLA MUNDO" casi en el centro y donde coloquemos el mouse se dibujara un circulo amarillo
×Si algo salió mal	No se va a crear nada y habrá un error de compilación.
Qué revisar si no funcionó como debía...	✓ Fíjate de haber copiado todo el código del ejemplo. ✓ ¡No puede haber fallado mucho más! No hay cables ni conexiones.

Y así como así, ¡creaste tu primer programa para una PC! Como puedes ver, es idéntico a como programamos Arduino salvo por algunas funciones que son específicas de cada plataforma. La función draw de Processing es equivalente a la función loop, por lo que todo el código que creemos allí se ejecutara una

y otra vez mientras el programa esté abierto. En Processing vamos a ser capaces de hacer dibujos en la pantalla utilizando un conjunto de funciones que permiten trazar desde puntos, líneas, cuadrados o círculos hasta imágenes.

La función background (que significa "fondo" en inglés) nos permite redibujar todo lo que está en la pantalla poniéndole un color determinado. Es posible dibujar un fondo de color rojo utilizando por ejemplo background(255,0,0). Vamos a ver cómo funciona esto: en el monitor, el color de un píxel (que son cada uno de los puntos que componen la imagen) está determinado por tres valores, uno para el color rojo, otro para el verde y otro para el azul. Como la composición del color aquí es aditiva (por luz), funciona diferente del color con pigmentos (por ejemplo pintando con témpera). Por una combinación de estos tres colores (rojo, verde, azul) podemos obtener cualquier color. Aquí algunos ejemplos más para entender mejor:

Ejemplo colores:

Blanco
Rojo 255
Verde 255
Azul 255

Amarrillo
Rojo 255
Verde 255
Azul 0

Negro
Rojo 0
Verde 0
Azul 0

Rojo
Rojo 255
Verde 0
Azul 0

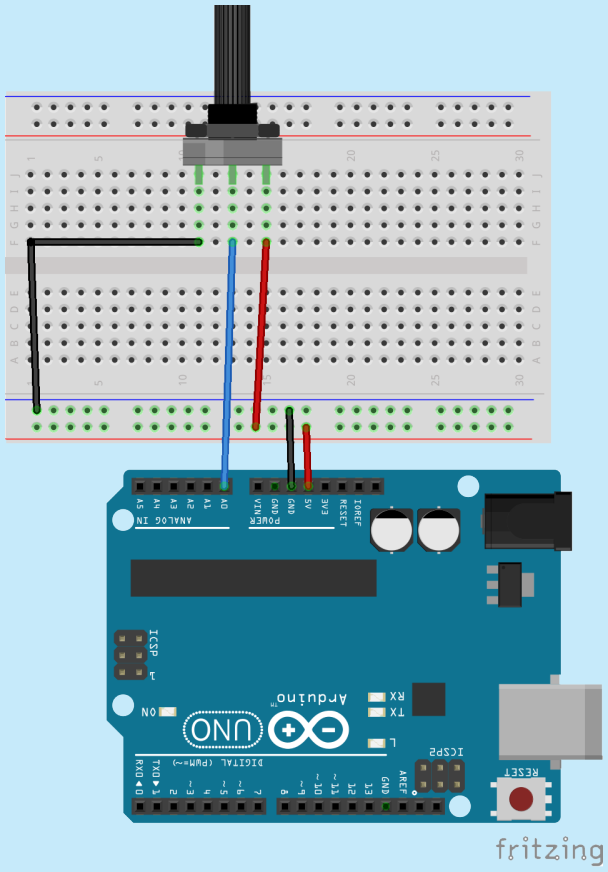
Celeste
Rojo 0
Verde 255
Azul 255

Violeta
Rojo 150
Verde 130
Azul 255

El resto de las funciones están explicadas con comentarios dentro del código. Hay mucho para descubrir en Processing, pero no es el objetivo principal de este material. ¡Sino no se terminaría jamás! Vamos a ver solo lo básico y pasaremos ahora a hacer una comunicación entre nuestra PC y otro programa que crearemos en Arduino. Lo primero es enviar el valor que lee un potenciómetro del Arduino para graficarlo en pantalla.

A continuación, crearemos el siguiente circuito en Arduino usando un potenciómetro solamente:

Esquema: Jugando con el servomotor



Y cargaremos este código:

Código Arduino: ComunicacionConPC01

```
void setup() {
  Serial.begin(9600); // creamos la conexión serial
  pinMode(A0,INPUT); // usaremos el pin analógico como entrada
}

void loop() {
  int valorLeido = analogRead(A0);// leemos el valor
  byte datoAEnviar = byte(map(valorLeido, 0, 1024, 0, 255)); // hacemos que entre en un byte
  // es decir transformaremos el valor que va de 0 a 1024 que leemos a
  // un valor que va de 0 a máximo 255
  Serial.write(datoAEnviar); // enviamos el valor en byte por el puerto serial
  delay(50); // dejamos pasar un par de milisegundos
}
```

Bien, ahora lo cargamos y lo probamos viendo el monitor serial. No te asustes, es normal que se vean caracteres raros cuando se mueve el potenciómetro, esto sucede porque estamos viendo un byte y el monitor serial lo quiere convertir en un carácter ASCII.

✓Si todo salió bien	En el monitor serial aparecerán las lecturas del potenciómetro con caracteres raros.
×Si algo salió mal	No vas a ver nada en el monitor serial.
 Qué revisar si no funcionó como debía...	✓ Fíjate que el puerto del monitor serial sea el correcto (debe ser el mismo que se usa cuando cargas los programas). ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos.

A continuación vamos a hacer un programa en Processing que lea este valor y que pinte el color de fondo con ese valor.

Código Processing: ComunicacionConArduino01

```
import processing.serial.*;

Serial miPuerto;
int valorLeido = 0;

void setup() {
  size(640, 480); // creamos la ventana
  miPuerto = new Serial(this,"COM5",9600); // creamos una conexión con serial con el
  // Arduino
  // en general el puerto es COM5 pero si no funciona hay que ver en el IDE Arduino qué puerto
  //está usando
}

void draw() {
 if (miPuerto.available()>0) { // si hay datos para leer en el puerto
     valorLeido = miPuerto.read(); // leemos el valor del que hay en el puerto
 }
 background(valorLeido,valorLeido,valorLeido); // dibujamos un fondo de negro a blanco dependiendo
 // del valor leído
}
```

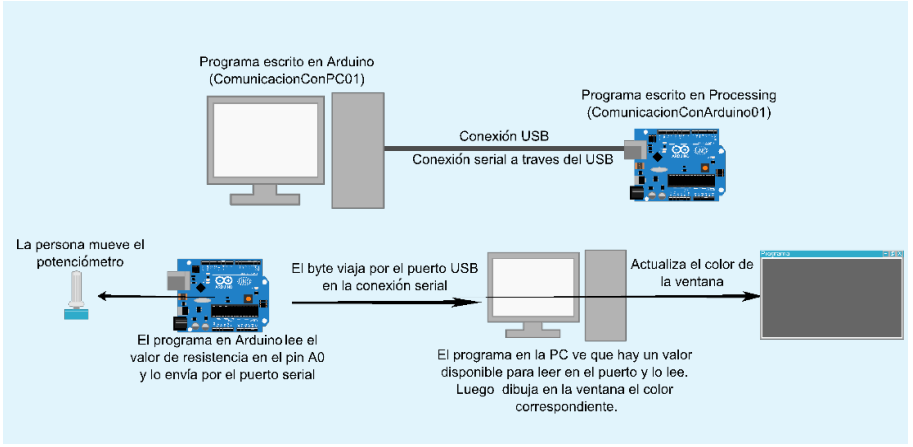
Bien, ahora compila ese código en el IDE Processing y ejecútalo mientras el Arduino está conectado a la PC con el USB.

✓Si todo salió bien	En el monitor debería crearse la ventana y al mover el potenciómetro debería pasar de blanco a negro, a través de grises.
×Si algo salió mal	Puede verse un error en el IDE Processing.
 Qué revisar si no funcionó como debía...	✓ Fíjate que el puerto del programa (por defecto “COM5”) sea el correcto. Debe ser el mismo que se usa cuando cargas los programas. ✓ Revisa que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB.

- ✓ Revisa el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos.
- ✓ Verifica que el Arduino esté conectado y tenga cargado el programa **ComunicacionConPC01.**

Al fin, ¡nuestro Arduino le envía mensajes a la computadora ejecutando nuestro programa escrito en Processing!

Vamos a ver cómo funciona todo esto con el siguiente esquema:

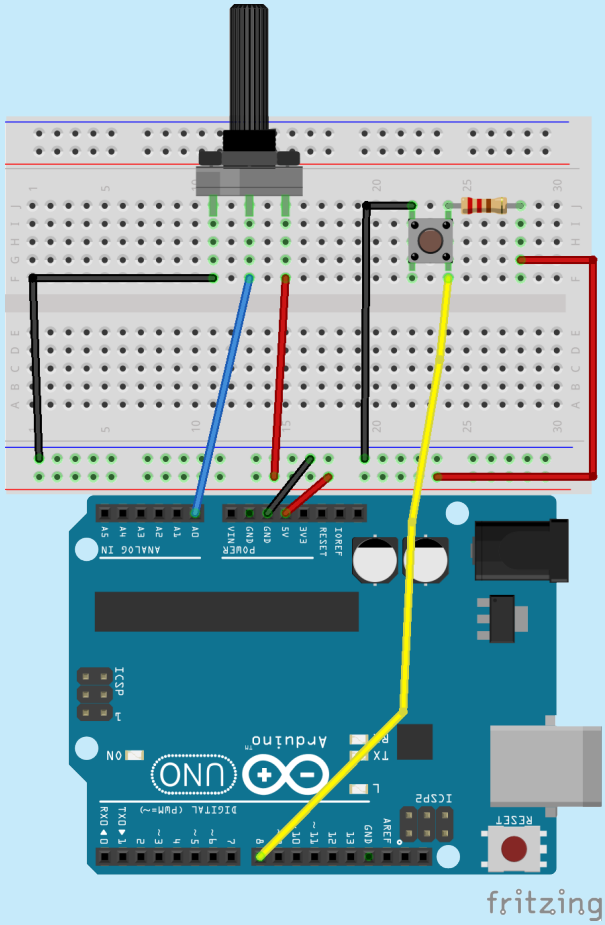


Ahora vamos a modificar nuestro programa para crear algo más que grises. Para eso tendríamos que cambiar individualmente cada uno de los componentes RGB (sigla en inglés para Rojo, Verde y Azul, por los tres valores de luz que puede tener un píxel). Una forma de lograrlo con un solo botón es hacer un selector que cada vez que se presione cambie el componente que estamos modificando. Es decir que si ahora estamos cambiando el color rojo, al presionar el botón pasamos al componente verde, luego al azul y luego volvemos al rojo.

Para lograr eso deberíamos hacer que nuestro Arduino le envíe dos valores distintos a Processing. Hay muchas formas de hacer esto, la de este ejemplo es una de las sencillas: lo que haremos es enviarlos en orden y poniendo un valor para comunicarle a Processing qué es lo que significa el valor que leerá a continuación. Antes de mandar el valor leído del potenciómetro mandaremos una “P”, y antes de mandar el valor del botón una “B”. Así sabremos desde Processing de qué se tratará el valor siguiente.

Vamos con los esquemas, los códigos y las pruebas.

Esquema: Jugando con el servomotor



Código Arduino: ComunicacionConPC02

```
void setup() {
  Serial.begin(9600); // comenzamos la conexión serial
  pinMode(A0,INPUT); // usaremos A0 para el potenciómetro como input
  pinMode(8, INPUT); // usaremos el pin 8 como input
}

void loop() {
  Serial.write('P'); // vamos a escribir el valor del potenciómetro
  Serial.write( byte(map(analogRead(A0),0,1023,0,255))); // leemos el potenciómetro

  Serial.write('B'); // vamos a escribir el valor del botón
  if (digitalRead(8)==HIGH) { // si está HIGH
    Serial.write('V'); // es verdadero, es decir se presionó
  } else {
 Serial.write('F'); // es falso, es decir no se presionó
  }

  delay(100); // esperamos 100 milisegundos
}
```

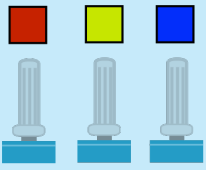
Código Processing: ComunicacionConArduino02

```
import processing.serial.*;

Serial miPuerto;

int valorLeido = 0; // el valor que leeremos del puerto
int potenciometro = 0; // el valor actual del potenciómetro
int valorR = 255; // el valor de color Rojo para el fondo
int valorG = 255; // el valor de color Verde para el fondo
int valorB = 255; // el valor de color Azul para el fondo
char seleccion = 'R'; // qué valor estamos cambiando ahora si R (rojo) G (verde) o B (Azul)

boolean presionoBoton = false;
boolean estadoAnterior = false;

void setup() {
  size(640, 360);
miPuerto = new Serial(this,'COM5',9600); // creamos una conexión con serial con el Arduino
  // en general el puerto es COM5 pero si no funciona, hay que ver en el IDE Arduino qué puerto está usando
}

void draw() {
 if (miPuerto.available()>0) { // si hay datos para leer en el puerto
     valorLeido = miPuerto.read(); // leemos el valor del puerto
     if (valorLeido=='P') { // si leo una "P" entonces el valor que sigue es el del potenciómetro
       potenciometro = miPuerto.read();
     } else {  // sino
       if (valorLeido == 'B') { // si es una "B" es el del botón
         presionoBoton = miPuerto.read() == 'V'; // si es "V" de verdadero dará true, sino false
       } else {// si es otra cosa leemos igual para vaciar el buffer
         miPuerto.read();
       }
     }
 }

 if (estadoAnterior!=presionoBoton && presionoBoton == true) {// si el estado del botón cambió, y está
   // presionado ahora debería cambiar el color que tenemos seleccionado para cambiar

     if (seleccion == 'R') { // si el color actual es R
       seleccion='G'; // pasamos al que sigue G
     } else { //sino
       if (seleccion == 'G') { // si es G
           seleccion='B'; // pasamos a B
       } else {
           if (seleccion == 'B') { // y si es B
             seleccion='R'; // volvemos a R de nuevo.
           }
       }
     }

 }

 if (seleccion == 'R') {// si la seleccion es R, cambiamos el valor R del color del fondo
   valorR=potenciometro;
   fill(255,0,0); // y dibujaremos el rectángulo de color rojo
 }
 if (seleccion == 'G') { // lo mismo con verde
   valorG=potenciometro;
   fill(0,255,0);
 }
 if (seleccion == 'B') { // lo mismo con azul
   valorB=potenciometro;
   fill(0,0,255);
 }

 background(valorR,valorG,valorB); // dibujamos el fondo del color que creamos
 rect(8,8,8,8);// dibujamos un pequeño rectángulo del color que corresponde
 fill(0,0,0); // vamos a dibujar ahora con negro el texto para ver cómo está formado el color de fondo
 text("Rojo: "+ valorR + " Verde: " + valorG + "Azul: " + valorB,24,16); // en sus componentesRGB
 estadoAnterior = presionoBoton; // actualizamos el estado del botón
```

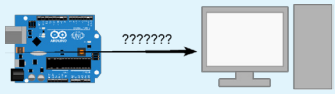
✓Si todo salió bien	En el monitor se creó la ventana y al mover el potenciómetro pasó de blanco a negro, a través de grises.
×Si algo salió mal	Puede verse un error en el IDE Processing.
 Qué revisar si no funcionó como debía...	✓ Fíjate que el puerto del programa (por defecto "COM5") sea el correcto. Debe ser el mismo que se usa cuando cargas los programas. ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos. ✓ Verifica que el Arduino esté conectado y tenga cargado el programa **ComunicacionConPC01.**

¡HORA DE EXPERIMENTOS Y EJERCICIOS!

Ahora que ya sabes un poco más sobre Arduino por qué no intentás...



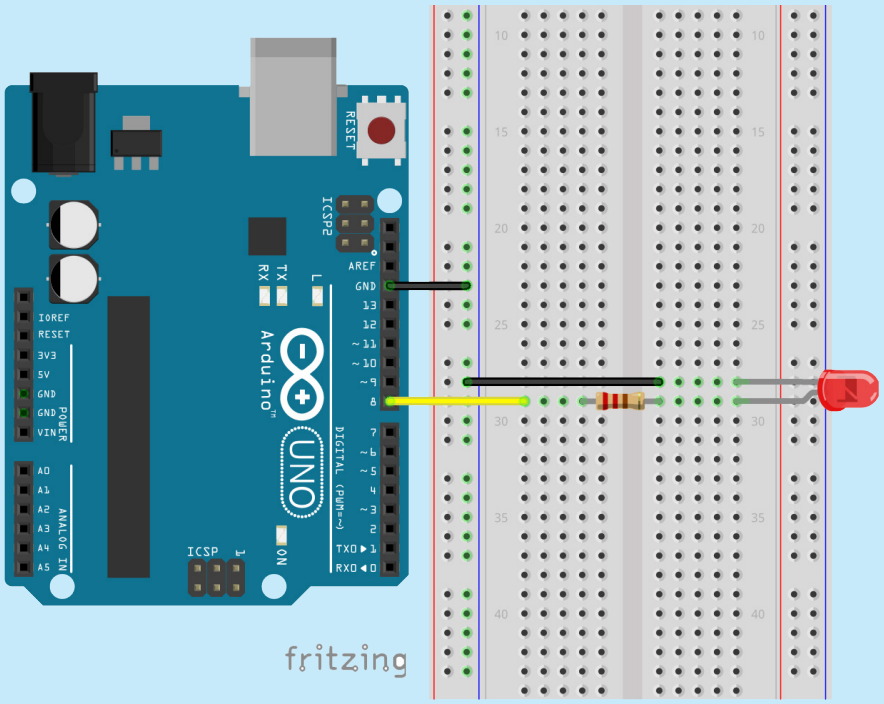
1. Agregar dos potenciómetros más y sacar el botón, así manejas cada componente RGB con un potenciómetro.



2. ¿Es necesario usar una letra para indicar a qué corresponde el valor? ¿De qué otra forma podrías mandar más de un dato? **Pista**: el orden importa.

Para comunicar en sentido inverso, es decir desde el Arduino a la PC, simplemente hacemos todo en sentido inverso. ¡Vamos a probarlo, haciendo que al presionar una el click del mouse, se encienda o apague un led en Arduino!

Esquema: OtraVezBlinkAhoraConlaPC



Código Arduino: ComunicacionConPC03

```
void setup() {
pinMode(8,OUTPUT); // usamos el pin 8 para el Led
digitalWrite(8,LOW);
  Serial.begin(9600); // iniciamos la comunicacion serial
}

void loop() {
if (Serial.available()>0) { // si hay datos para leer
    byte valorLeido = Serial.read(); // leerlo
    if (valorLeido=='1') { // si es un 1
        digitalWrite(8,HIGH); // prendemos el led
    } else {
        digitalWrite(8,LOW); // sino, lo apagamos
    }
  }
}
```

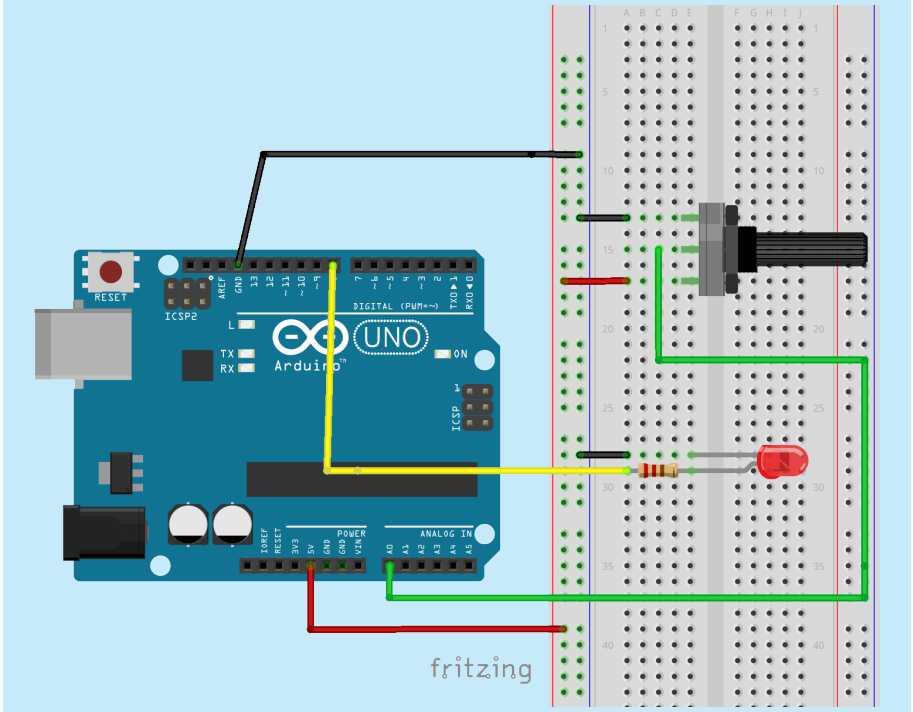
Código Processing: ComunicacionConArduino03

```
import processing.serial.*;

Serial miPuerto;
boolean estadoAnterior = false;

void setup() {
  size(640, 360);
miPuerto = new Serial(this,"COM5",9600); // creamos una conexion con serial con el Arduino
  // en general el puerto es COM5 pero si no funciona, hay qué ver en el IDE arduino que puerto
// está usando
}

void draw() {
  if (mousePressed == true) { // si se hizo click
if (estadoAnterior != mousePressed) {   // y antes no se estaba haciendo click
      miPuerto.write('1'); // enviamos un 1 a Arduino
    }
  } else { // si no se hizo click
    if (estadoAnterior != mousePressed) {  // y antes SI se estaba haciendo click
      miPuerto.write('0'); // enviamos un 0 a Arduino
    }
  }

  estadoAnterior = mousePressed;
}
```

✓Si todo salió bien	Cuando hagamos clic en la pantalla de la PC, en la ventana de nuestro programa Processing, el led se encenderá en el Protoboard.
×Si algo salió mal	Puede verse un error en el IDE Processing o no prenderse nunca el led.
 Qué revisar si no funcionó como debía...	✓ Fíjate que el puerto del programa (por defecto "COM5") sea el correcto. Debe ser el mismo que se usa cuando cargas los programas. ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos. ✓ Verifica que el Arduino esté conectado y tenga cargado el programa **ComunicacionConPC03.** ✓ Verifica estar ejecutando también el programa Processing **ComunicacionConArduino03.**

Queda una última pregunta. ¿Podemos enviar muchos datos del Arduino a la PC? Y lo mismo pero al revés, ¿cómo hacemos para enviar y recibir datos? Para eso tenemos que hacer lo que se llama un serial "handshake" (apretón de manos o saludo). Se usa para que uno de los dos programas espere al otro para comenzar la comunicación y hacerla ordenada. Veamos el siguiente esquema:

Esquema: UltimoEsquemaSerial



Código Arduino: ComunicacionConPC04

```
char valorLeido;  //almacenamos el valor leído

void setup()
{
  pinMode(8, OUTPUT); // usamos el led en el pin 8
  Serial.begin(9600); // iniciamos la comunicacion a 0600 bps
  esperarSaludo();  // enviamos saludos, hasta recibir uno
}

void loop()
{
  if (Serial.available() > 0) { // si hay datos
    valorLeido = Serial.read(); // lo leemos y guardamos en valor leído
    if(valorLeido == '1') { // si se presiona el botón del mouse, el valor leído es 1
       digitalWrite(8, HIGH);  // encendemos el led
    }
    if(valorLeido == '0') { // si es 0 entonces no se presiona el mouse
       digitalWrite(8,LOW);// apagamos el led
    }
  } else {
    int potenciometro = analogRead(A0);// leemos el valor
    byte datoAEnviar = byte(map(potenciometro, 0, 1024, 0, 255));// lo ponemos entre 0 y 255
    Serial.write(datoAEnviar); // lo enviamos
  }
  delay(100);
}

void esperarSaludo() { // esta función espera un saludo
  while (Serial.available() <= 0) {//mientras no haya respuesta repetir el siguiente código
    Serial.write("A");   // enviamos el saludo 'A'
    delay(300);// esperamos
  }
}
```

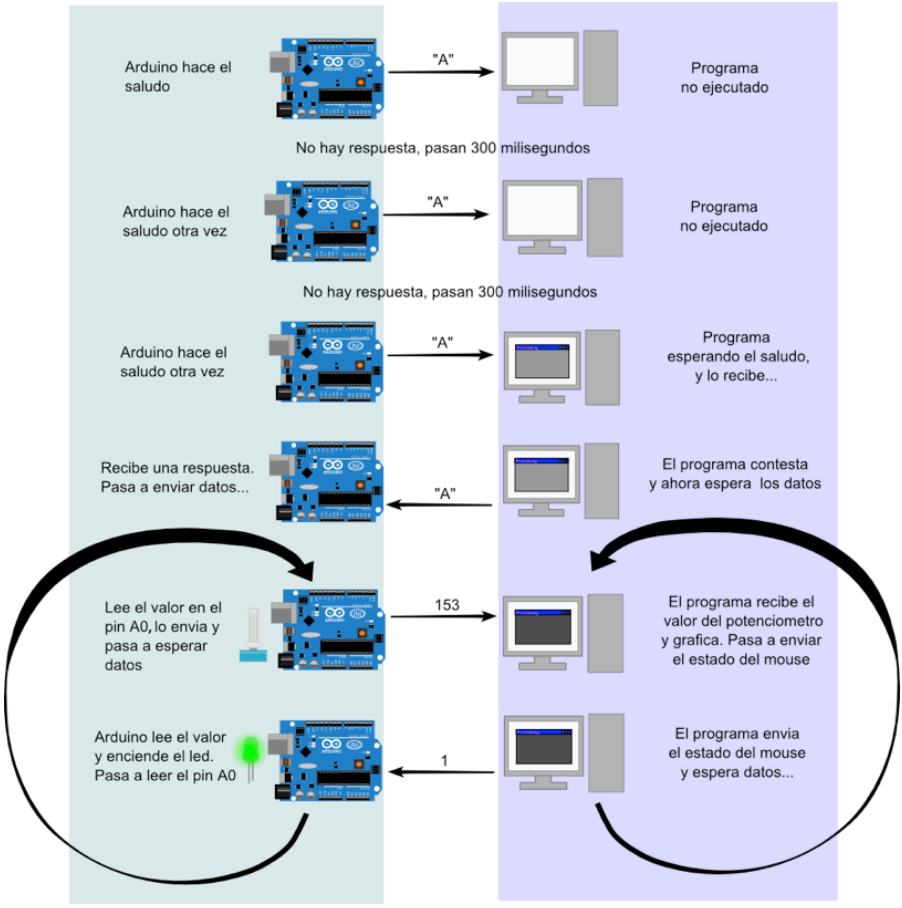
Código Processing: ComunicacionConArduino04

```
import processing.serial.*;

Serial miPuerto;
int valorLeido;


void setup() {
size(200, 200); // usaremosunaventanapequeña.
// miPuerto = new Serial(this,"COM5",9600); // recuerda cambiar el Puerto al que
// corresponda en tu pc
}

void draw() {
  if (miPuerto.available()>0) {
valorLeido = miPuerto.read();
      if (primerSaludo == false) {// si no estableció el primer saludo
        if (valorLeido == 'A') {// si el valor de val es A, entonces el Arduino nos
está saludando.
          miPuerto.clear(); // limpiamos el estado del puerto
          primerSaludo = true; // ahora sí se estableció el primer contacto
          miPuerto.write('A'); // escribimos una A para saludar al Arduino
        }
      } else { //si ya se había hecho el primer contacto
        // hacer algo con el valor leído

        if (mousePressed == true) {// si el mouse está presionado mandar un 1
miPuerto.write('1');        //send a 1
        } else {
miPuerto.write('0'); // sino un 0
        }
    }
  }
delay(100);// esperamos
background(valorLeido,valorLeido,valorLeido); // dibujamos el fondo
}
```

✓Si todo salió bien	Cuando hagamos clic en la pantalla de la PC, en la ventana del programa Processing, el led se encenderá en el protoboard, y al tocar el potenciómetro en el Arduino, cambiará el color de la pantalla.
×Si algo salió mal	Puede verse un error en el IDE Processing. No prenderse nunca el led. No recibir datos el programa Processing.
 Qué revisar si no funcionó como debía...	✓ Fíjate que el puerto del programa (por defecto “COM5”) sea el correcto. Debe ser el mismo que se usa cuando cargas los programas. ✓ Revisar que el código haya sido cargado al Arduino y que este se encuentre conectado al puerto USB. ✓ Revisar el conexionado varias veces. Hay muchos cables conectados, fijarse que no estén sueltos. ✓ Verifica que el Arduino esté conectado y tenga cargado el programa **ComunicacionConPC04**

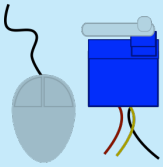
✓ Verifica estar ejecutando también el programa Processing **ComunicacionConArduino04.** Primero debe correr el programa en el Arduino y luego en la PC.

Bien, ahora sí tenemos comunicación en ambos sentidos. Lo que hacen ambos programas, básicamente, es turnarse para enviar datos. Para poder hacerlo deben sincronizarse para saber cuál comenzará primero. En lugar de jugar piedra papel o tijera y decidirlo a la suerte, uno de los programas siempre enviará el saludo (en nuestro caso Arduino) y el otro siempre esperará este saludo. Una vez que se han saludado, ya sí pueden turnarse para enviar y recibir. Veámoslo mejor en el siguiente esquema:

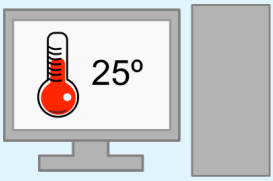


¡MÁS EXPERIMENTOS Y EJERCICIOS!

Ahora que ya sabes un poco más sobre Arduino por qué no intentás...



1. ¡Manejar un motor en el Arduino usando el mouse de la PC!



2. Hacer que el termómetro del proyecto 3 envíe el dato a la computadora y graficarlo con Processing de la manera que se te ocurra.

CONCLUSIÓN

Un abanico de posibilidades: ¿hacia dónde seguir?

A través de este material hemos descubierto y trabajado con algunos elementos propios de la electrónica, de la programación y del mundo Arduino que permitieron desarrollar experiencias propias de mediana complejidad. Desde el primer proyecto, el semáforo, hasta el último, donde se establece una comunicación entre el Arduino y la computadora personal, se ha experimentado de forma integral con los diferentes tipos de módulos (sensores y actuadores), así también como con las diferentes lógicas de control necesarias para hacer uso de ellos mediante la creación de programas. Esto ha permitido desarrollar un abanico de posibilidades que serán útiles a la hora de crear proyectos nuevos de lo más variados.

Lo que hemos explorado hasta aquí, sin embargo, no representa de forma completa a cada una de estas disciplinas. Por el contrario, es solo una introducción práctica, didáctica e integrada de todas ellas. Estos elementos iniciales habilitan las herramientas necesarias para emprender exploraciones aún más profundas en el mundo de la programación, de la electrónica o de la fusión entre ambas, es decir en el desarrollo de aparatos interactivos digitales.

El carácter multidisciplinario que promueve y requiere el trabajo con Arduino (y también otros microcontroladores similares) hace que sea más fácil trabajar con elementos de estas disciplinas y verlas al mismo tiempo en acción e interacción. Sin embargo, no existe mucha oferta de material que, como el actual, se dedique a trabajarlos de forma simultánea. Por eso, para poder profundizar los conocimientos en estas áreas, será necesario complementar este material con fuentes de información específicas de cada una y acompañar esta exploración con experiencias en proyectos prácticos cada vez más complejos.

Además, la posibilidad de trabajar con sensores y actuadores del mundo real implica la oportunidad de experimentar y jugar, por qué no, con elementos que escapan a estas disciplinas. Los engranajes, los motores y los movimientos mecánicos en general hacen necesario el uso de conocimientos en física; la posibilidad de contar con sensores de gases, de temperatura y de humedad permiten también emprender proyectos en relación con la química; la capacidad de crear sonidos, imágenes y movimientos también permite ex-

plorar las distintas ramas del arte, ya sean visuales o auditivas; y por último, crear objetos interactivos útiles supone contar con una cierta perspectiva social y reflexionar sobre qué proyectos pueden ayudar a una institución, a una persona o la comunidad en general.

El desarrollo de habilidades que tengan que ver con la creación de elementos digitales permite, además, cambiar la forma en que se conciben dichas tecnologías. Estos aparatos pueden dejar de verse como cajas negras, objetos mitificados o sagrados, y pasar a interpretarse simplemente como objetos diseñados para cumplir una determinada tarea. Conocer cómo funcionan permite tener la posibilidad de intervenirlos para adaptarlos a nuestras necesidades y circunstancias particulares.

En un mundo donde las tecnologías digitales dominan e influyen en gran parte de nuestra vida social y económica, es importante tener una visión más profunda que permita hacer un uso creativo y crítico de ellas. Tener los conocimientos necesarios para intervenirlas nos permite escapar a los usos guionados que proponen algunas corporaciones, en virtud de perseguir la libertad y soberanía sobre nuestras decisiones y caminos.

Anexo
Guía de programación

SECCIÓN 1

El lenguaje de programación

Al microcontrolador se lo programa por medio del lenguaje de programación Arduino (el cual está basado en Wiring)[1] usando un entorno de desarrollo propio (IDE Arduino) que puede descargarse de la página oficial.[2] El IDE se ejecuta en una computadora normal y es allí donde se escriben los programas, que luego son cargados a la placa por medio de un cable USB. Todo programa puede ejecutarse en la placa Arduino sin necesidad de estar conectado a una computadora.

El lenguaje Arduino es de tipo imperativo o por procedimientos, lo cual significa que la programación se construye como un conjunto de instrucciones que se ejecutan paso a paso y le indican a la computadora (en forma de órdenes) cómo realizar una tarea. Es bastante sencillo de aprender, ya que desde su diseño fue pensado para ser utilizado por un público amplio. Por otro lado, existe una comunidad importante de personas haciendo cosas con Arduino y compartiéndolas, como así también colaborando con dudas o problemas en foros de Internet.

- **Foros oficiales Arduino de apoyo**
  http://forum.arduino.cc/index.php
- **Club Arduino Argentina**
  https://www.facebook.com/clubarduinoargentina
- **Foro ArduinoRed de Escuelas**
  http://proyectoarduino.foroactivo.com/f1-foro-proyecto-arduino
- **Foro ArduinoSpain Labs**
  http://www.spainlabs.com/foro/viewforum.php?f=9

1. Para más información, se puede consultar la página oficial de Wiring (http://wiring.org.co/) o el artículo de Wikipedia sobre esta plataforma (http://goo.gl/dCBwp0). Ambos están en inglés.

2. Página oficial de Arduino: http://arduino.cc/en/Main/Software

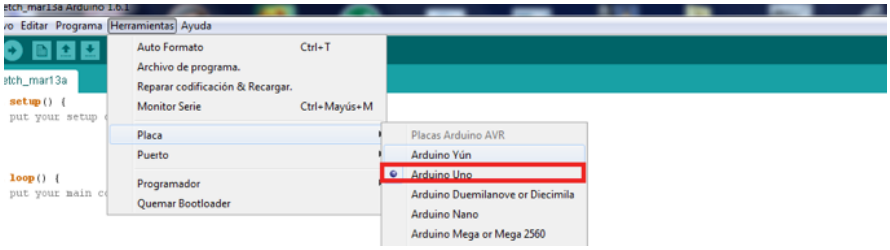
SECCIÓN 2

El entorno de desarrollo de Arduino

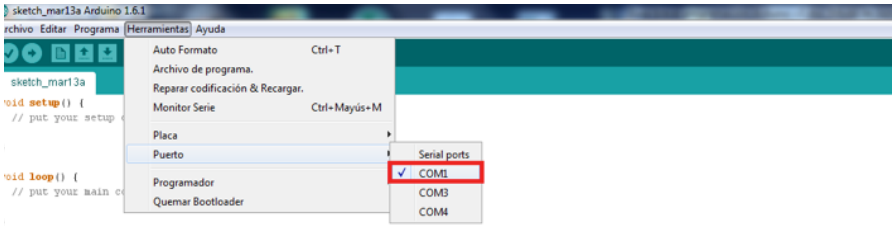
El IDE se puede obtener desde la sección de descargas (Download) en la página oficial de Arduino. Se encuentra disponible para distintos sistemas operativos: Windows, Mac y Linux.

Una vez instalado el entorno IDE, hay que realizar dos tareas antes de comenzar a programar:

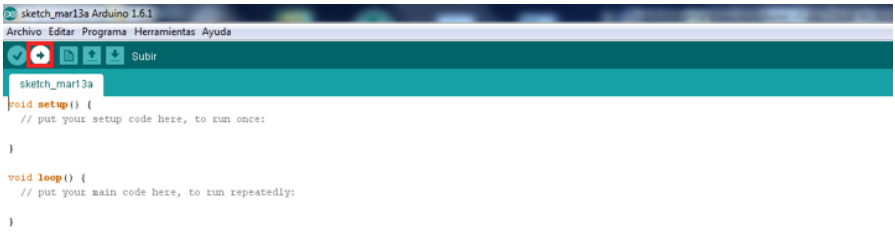
a) Indicar en el menú el modelo de placa Arduino con que se cuenta (el más utilizado es el Arduino UNO v3).



b) Configurar la comunicación entre la placa y la computadora. En el entorno, se debe ir al menú Tools, opción Serial Port. Luego, seleccionar el puerto serie al que está conectada nuestra placa. En Windows, este dato se obtiene a través del software del sistema operativo Administrador de dispositivos, opción Puertos COM & LPT/USB Serial Port.



A partir de este momento ya puedes copiar en el editor IDE y luego probar los programas de esta guía. Recuerda que a un programa se lo transfiere a la placa y se ordena su ejecución con la instrucción:



SECCIÓN 3

Conceptos básicos

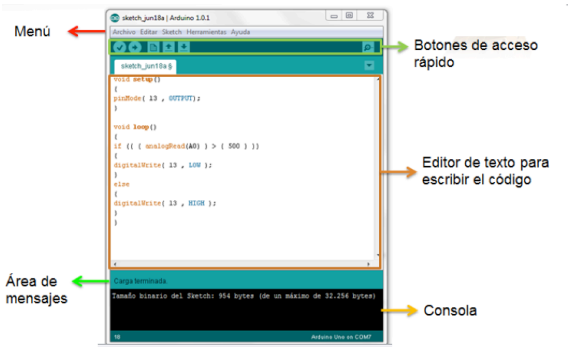
Antes de empezar a realizar las primeras prácticas de programación es necesario presentar una serie de conceptos básicos que ayudaran a entender cómo se construye un programa. Luego de este apartado, recién en el punto 4 y una vez integrando lo visto, podremos copiar al editor un programa completo, transferirlo a la placa Arduino y verificar su ejecución.

Por cualquier ayuda complementaria, se recuerda que la guía oficial de programación de la placa Arduino se encuentra disponible en Internet.

3.1. PROGRAMA

Un programa es un conjunto de órdenes o instrucciones escritas en un lenguaje artificial, que una computadora puede entender y ejecutar, y que resuelven un problema específico. Un programa está escrito en instrucciones propias de un lenguaje de programación (por ejemplo BASIC, C, Java, etc.), las cuales incluyen un conjunto de reglas, notaciones, símbolos y/o caracteres. En el ambiente Arduino a los programas se los llama también "sketchs".

En la imagen siguiente observamos el IDE de trabajo, que se ejecuta en una computadora asociada a la placa Arduino por medio de un cable USB. En el marco central se muestra un programa que se está construyendo (en la jerga informática se dice "editando"). Las flechas indican las distintas secciones que conforman la interfase de trabajo.



Ejemplo de una porción de programa editado en el IDE Arduino.

3.2. VARIABLE

Una variable es una forma de nombrar a un determinado valor que se desea almacenar para usarlo posteriormente dentro de un programa. En otras palabras, una variable es como una especie de "caja" que puede contener un objeto determinado (un número, un valor de verdad, una palabra o una frase). Como su nombre lo indica, lo que almacena o contiene puede cambiar varias veces en la ejecución de un programa.

Al definir o crear una variable se le debe dar un nombre para poder utilizarla luego. Para facilitar la lectura del programa, este debe ser representativo de su contenido (por ejemplo "saldo", "apellidoCliente", "nombreCliente").

Las variables tienen un ámbito de aplicación (espacio de un programa) donde pueden utilizarse. A las variables que se pueden utilizar en todo el programa se las denomina variables globales, mientras que las restringidas a partes específicas son variables locales, ya que solamente son accesibles desde dentro del espacio donde fueron creadas.

Una variable debe ser declarada y opcionalmente se le asigna un determinado valor. En la declaración de la variable se indica el tipo de datos que almacenará (int, float, long, etc)

En la programación se pueden definir otro tipo de variables de tipo constante, cuyo contenido asociado no cambia a lo largo de la ejecución del programa. Se usa la instrucción #define, la cual permite que el programador dé un nombre a un valor constante antes de que se ejecute el programa.

Ejemplos:

Se define la variable de nombre "inputVariable" como numérica, de tipo entero, y se almacena un cero en ella.

Fragmento de código

```
int inputVariable = 0;
```

Se define la variable lógica "estaConectado" y se almacena el valor falso.

Fragmento de código

```
boolean estaConectado = false;
```

Se define la variable de caracteres "cartelPeligro" y se almacena una frase.

Fragmento de código

```
String cartelPeligro = "Control detenido por calor";
```

Se define la constante PinLed.

Fragmento de código

```
#define PinLed 3
```

Programa ejemplo donde se muestran distintos tipos de datos, variables y ámbitos.

Código: primerPrograma

```
#define PinLed 3
String cartelPeligro = "Control detenido"; // cartelPeligro variable global
void setup() {
   Serial.begin(9600); // comienza la conexión serial
}
void loop() {
     int x = 5000; // x variable local
Serial.println(cartelPeligro); // imprime por serial la variable global
     digitalWrite(PinLed,HIGH); // enciende el led en el pin definido por la constante
delay(x); // espera 5000 milisegundos antes de continuar (lo que vale x)
}
```

3.3. TIPOS DE DATOS BÁSICOS

Arduino permite manejar los siguientes tipos de datos asociados a variables:

Tipo	Descripción
byte	Almacena un valor numérico de 8 bits. Tienen un rango de 0 a 255.
int	Almacena un valor entero de 16 bits con un rango de 32.767 a 32.768.
long	Valor entero almacenado en 32 bits con un rango de 2.147.483.647 a -2.147.483.648.
boolean (o lógico)	Almacena un valor verdadero (true) o falso (false).
String	Almacena una cadena alfanumérica, por ejemplo “hola mundo”
float	Almacena un valor numérico con decimales.

3.4. OPERADORES

Un operador es un carácter, o una secuencia de caracteres (+, -, >=, <=), que define una operación a realizar sobre un dato.

Por ejemplo, el operador matemático suma (+) se puede aplicar sobre la variable i de la siguiente forma: i = i + 1. Al valor actual de la variable i se le suma 1 y el resultado se almacena en i.

3.4.1. DE COMPARACIÓN

Operador	Descripción
==	x == y, ¿es x igual a y?
!=	x != y, ¿es x distinto de y?
<	x < y, ¿es x menor que y?
>	x > y , ¿es x mayor que y?
<=	x <= y , ¿es x menor o igual que y?
>=	x >= y , ¿es x mayor o igual que y?

3.4.2. OPERADORES ARITMÉTICOS

Operador	Descripción
=	a = b , asignación
+	a + b , adición o suma
-	a - b , sustracción o resta
*	a * b , multiplicación
/	a / b , división
%	a % b , módulo (resto de la división)

3.4.3. OPERADORES COMPUESTOS

Operador	Descripción
++	a++ , incremento (suma 1)
--	a-- , decremento (resta 1)
+=	a += b , adición compuesta
-=	a -= b , sustracción compuesta

3.4.4. OPERADORES BOOLEANOS

Operador	Descripción
!	!a, negación
&&	a && b, “y” lógico
||	a || b, “o” lógico

Aparte de los operadores booleanos, el lenguaje de Arduino presenta las siguientes constantes predefinidas TRUE (verdadero) / FALSE (falso).

3.5. FUNCIONES

Una función es un bloque de instrucciones, identificadas por un nombre, que realizan una determinada tarea dentro de un programa. En general las fun-

ciones son utilizadas para realizar tareas repetitivas. Por ejemplo, en un programa de contabilidad una función que construye un programador podría ser "calcularSaldo()", dado que esta tarea va a ser utilizada en distintas partes.

En un programa una función es ejecutada a partir de una llamada que involucra su nombre.

Para escribir una función primero se debe especificar el tipo de datos que devuelve (por ejemplo "int" si lo que devuelve es un valor entero) y luego, entre llaves, se escribe el cuerpo de la función (las instrucciones que la componen). Cuando la función devuelve un valor esto se hace mediante la instrucción "return".

En el siguiente ejemplo se muestra la estructura de una función típica:

Fragmento de código

```
// función multiplicar, devuelve un valor entero
int multiplicar(int a, int b) //a y b son dos variables que recibe la función
// multiplicar es el nombre de la función
{
        int resultado; // se declara una variable local
        resultado = a * b; // se multiplican los parámetros y se guardan en resultado.
        return resultado; // es el dato que devuelve la función
}
```

Por otro lado existen una serie de funciones que vienen con el lenguaje Arduino, es decir que las podemos usar directamente en nuestros programas, por ejemplo:

Funciones matemáticas

Función	Descripción
min(a,b)	devuelve mínimo entre a y b
max(a,b)	devuelve máximo entra a y b
abs(a)	devuelve valor absoluto de a
pow(a,b)	devuelve a^b (ambos tipo float), potenciación
sqrt(a)	devuelve la raíz cuadrada de a
sin(a)	devuelve el seno de a (a tipo float y en radianes)
cos(a)	devuelve el coseno de a (a tipo float y en radianes)
tan(a)	devuelve la tangente de a (a tipo float y en radianes)

Funciones de textos

Función	Descripción
txtMsg.**charAt**(4)	devuelve el carácter en la posición 4
txtMsg.**setCharAt**("A", 4)	sustituye el carácter en la posición 4 por "A"
texto1.**concat**("texto2")	concatena texto1 y texto2 (idéntico a texto1=texto1+texto2;)

txtMsg.**length**()	devuelve la longitud de la variable de texto
txtMsg.**toLowerCase**()	devuelve la cadena convertida en minúsculas
txtMsg.**toUpperCase**()	devuelve la cadena convertida en mayúsculas
texto1.**compareTo**(texto2)	compara dos cadenas. Devuelve 1 si texto1 es mayor que texto2, 0 si son iguales, y -1 en caso contrario
texto1.**equals**(texto2)	compara si dos cadenas son iguales (idéntico a texto1==texto2)
texto1.**equalsIgnoreCase**(texto2)	compara si dos cadenas son iguales, ignorando mayúsculas y minúsculas
txtMsg.**substring**(3, 10)	devuelve una subcadena de la posición 3 a la 10
txtMsg.**startsWith**("texto", 3)	comprueba si la cadena empieza por "texto", con offset 3
txtMsg.**indexOf**('A', offset)	devuelve el índice de la primera ocurrencia de 'A', a partir de la posición offset
txtMsg.**lastIndexOf**('A', offset)	devuelve el índice de la última ocurrencia de 'A' previa a la posición offset
txtMsg.**replace**("texto1", "texto2")	sustituye las ocurrencias de "texto1" por "texto2"

SECCIÓN 4

Estructura de un programa

Un programa Arduino es bastante simple desde su estructura. Esta se divide en dos partes, definidas por las funciones setup y loop.

- En la función **setup()** se insertan todas las instrucciones que ayudan a configurar un programa. Cuando un programa se ejecuta se llama a esta función como primera tarea, automáticamente y por una única vez.
- La función **loop()** contiene las instrucciones que componen a la parte principal del programa, es la ejecución. Cuando un programa Arduino se ejecuta el contenido de la función loop es ejecutado permanentemente de forma cíclica.

El siguiente programa es un ejemplo operativo (se puede copiar y probar sobre una placa Arduino) que muestra todos los conceptos explicados hasta el momento:

Código: ejemplo programa

```
int a = 1; // definición de variables globales
int b = 2;
void setup() { // función de configuración
    Serial.begin(9600); // se habilita comunicación con la PC para mostrar resultados
}

void loop()
{
  int mostrar; // definición de variable local
  mostrar = multiplicar(a, b); // llamada a función
  Serial.println(mostrar); // se transmite el resultado a la PC
  a = a + 1;
  delay(1000); // se incluye una espera de 1 segundo
}
// función multiplicar, devuelve un valor entero
int multiplicar(int a, int b) //a y b son dos variables que recibe la función
                                    // multiplicar es el nombre de la función
{
         int resultado; // se declara una variable local
         resultado = a * b;
         return resultado; // es el dato que devuelve la función
}
```

Como se observa en el programa anterior, cada instrucción finaliza con un punto y coma (“;”) y los comentarios del programador se indican con “//”, aunque también se los puede hacer en bloques con “/* texto de comentario */”.

SECCIÓN 5

Estructuras de bucles o ciclos

Las estructuras en bucle (*loops*) realizan una tarea de manera repetitiva hasta que una condición determina el fin del ciclo. En principio, presentamos dos estructuras de bucle diferentes: for y while.

5.1. FOR

La instrucción for (para) se usa para crear un bloque de instrucciones que se va ejecutar en un programa una cantidad determinada de veces. Un ejemplo de uso es el siguiente:

Fragmento de código

```
for(int i=1; i<=6; i++)
{
        j = j*i;
        println(j);
}
```

Nótese que la estructura usa una variable de iteración i, que es inicializada con el valor 1. Al final de cada ciclo el valor de la variable se incrementa en 1 (dado que se usa el operador "i++", el cual es equivalente a "i = i + 1"). Las instrucciones del interior del ciclo se ejecutan una vez tras otra hasta que i alcanza el valor 6. En ese caso particular ciclo finaliza, debido a que la condición de fin de ciclo "i<=6" cuando se evalúa da por resultado verdadero.

La primera línea de la instrucción for posee tres partes: inicialización (declaración de una variable contador, en este caso i), test o comprobación (en el ejemplo, i menor o igual a 6) e incremento (o decremento) de la variable de control.

La inicialización sucede una vez, solo al inicio del ciclo. El test o control se realiza en cada iteración del bucle. Si el resultado del test es verdadero (true), el bloque de código interior se ejecuta y el valor de la variable contador se incrementa (++) o disminuye (-) en una unidad. Por el contrario, si el resultado es falso (false), finaliza su ejecución y el programa continua con la instrucción inmediatamente posterior al ciclo.

Existen otras formas de incrementar, las cuales trabajan con otros valores distintos a 1. Para ello se utilizan los operadores de suma y resta compuesta "+=" y "-=" (por ejemplo, "i+=5" incrementa de a saltos de 5 unidades).

En el siguiente ejemplo se muestra el uso de for en un ciclo que primero calcula y muestra a un número multiplicado por sí mismo, luego lo incrementa en cada vuelta en uno, y finaliza cuando su valor llega a 12; en otro ciclo for, se realiza lo inverso. El ejemplo está completo, por lo cual se puede copiar y probar su funcionamiento en la placa.

Código: ejemploFor

```
void setup()
{
   Serial.begin(9600);
}
void loop() {
   for (int i = 0; i <= 12; i++){
     Serial.print("i: ");
     Serial.println(i*i);
   }
   for (int j = 12; j >= 0; j--){
     Serial.print("j: ");
     Serial.println(j*j);
   }
   delay(1000);
}
```

5.2. WHILE

El while (mientras) también se usa para crear un bloque de instrucciones que repetirá su ejecución mientras se cumpla con una condición lógica.

Un ejemplo de uso es el siguiente, donde las instrucciones contenidas en la estructura while se ejecutarán mientras la variable contador sea menor o igual a quince:

Fragmento de código

```
while(contador <= 15)
{
        instrucciones
}
```

En la declaración de la condición se utilizan los operadores lógicos (AND/Y, OR/O, NOT/NO). El siguiente es un ejemplo de un programa completo en el que se usa la instrucción while:

Fragmento de código

```
int contador=0;
void setup()
{
   Serial.begin(9600);
}
void loop()
{
  while(contador <= 10)
  {
    Serial.println(contador);
    contador++;
  }
}
```

SECCIÓN 6

Condicionales

Las estructuras condicionales son las sentencias utilizadas para decirle al programa que haga una cosa u otra. El lenguaje de Arduino permite realizar sentencias condicionales “if” e “if... else”.

6.1. IF

Cuando se quiere que el programa haga algo dependiendo de alguna condición se utilizara la instrucción if (si, condicional). La instrucción if verifica la condición de una expresión, escrita entre paréntesis, y si la evaluación de esta condición es verdadera se ejecutan las instrucciones estén contenidas en el bloque.

Fragmento de código

```
if (expresión)
{
        Hace esto;
        y esto también;
}
```

La expresión puede ser de los siguientes tipos:

- a == b (a es igual a b)
- a != b (a no es igual a b)
- a < b (a es menor que b)
- a > b (a es mayor que b)
- a <= b (a es menor que o igual a b)
- a >= b (a es mayor que o igual a b)

6.2. IF ... ELSE

Es una instrucción que trabaja de forma semejante a if, dado que evalúa una condición. En caso de que el resultado sea verdadero, se ejecuta el bloque si-

guiente de instrucciones (contenido primero en if) y en caso de que sea falso, se ejecuta solo el segundo bloque else (sino). Es un ejemplo completo, así que se recomienda su copia y prueba en la placa.

Código: contadorDePares

```
int contador=0;
void setup()
{
   Serial.begin(9600);
}
void loop()
{
  Serial.print(contador);
  if((contador % 2) == 0) // operador de módulo
  {
    Serial.println(" es par");
  }  else
  {
    Serial.println(" es impar");
  }
  contador++;
}
```

SECCIÓN 7

Otras instrucciones de control de flujo

7.1. RETURN

Return es una instrucción que se inserta en una función y que devuelve un valor al finalizar la ejecución de la misma.

Fragmento de código

```
// función multiplicar, devuelve un valor entero
int multiplicar(int a, int b)
{
         int resultado;
         resultado = a * b;
         return resultado; // es el dato que devuelve la función
}
```

7.2. BREAK

Instrucción que al ejecutarse rompe la iteración del bucle donde está inserta, saliendo del mismo sin tener en cuenta que se cumplan las condiciones marcadas para su fin. Es una instrucción cuyo uso debe tratar de evitarse, dado que puede complicar la programación.

Fragmento de código

```
int i = 0;
for ( ; ; ) {
          if (i < 10) {
                    Serial.println(i);
          } else {
                    break;
 }
 i++;
}
```

SECCIÓN 8

Funciones útiles

El lenguaje de programación de Arduino posee una serie de funciones que pueden auxiliar de manera significativa al programador, aliviando su tarea, dado que implementan rutinas ya programadas.

8.1. DELAY(M)

Cuando se inserta la instrucción delay en un programa, en el lugar que se la dispone se produce una pausa de ejecución de m milisegundos.

Fragmento de código

```
delay(1000);
```

8.2. PINMODE

Instrucción usada para configurar los pines de la placa Arduino, ya sea como entrada o salida. Se declarará dentro de la función setup, por lo que la configuración de pines solo se hará una vez, antes de empezar a ejecutar el programa principal.

Fragmento de código

```
pinMode (13,OUTPUT); //establece el pin 13 como salida
pinMode (15,INPUT); //establece el pin 15 como entrada
```

8.3. DIGITALWRITE Y DIGITALREAD

Las entradas y salidas digitales trabajan únicamente con dos estados, alto (HIGH) o bajo (LOW), los cuales irán asociados a un nivel de 5 voltios (alto) o de 0 voltios (bajo). De manera alternativa podemos encontrarnos con placas Arduino que trabajan a 3,3 voltios.

Fragmento de código

```
entrada = digitalRead(10); //asocia la variable "entrada" al pin digital 10
digitalWrite (10, HIGH); //establede el pin 10 con estado HIGH
```

8.4. ANALOGWRITE Y ANALOGREAD

El conjunto de pines de tipo analógico, a diferencia de los pines digitales, puede tomar cualquier valor entre 0 y 5 voltios. Es por ello que en el momento de leer un valor se tendrá una amplitud correspondiente a un rango de 10 bits o 1024 estados posibles (de 0 a 1023). Para escribir un valor analógico se usa la técnica PWM (por las siglas en inglés de Modulación por Ancho de Impulso) y para esto la resolución es de 8 bits (un rango de 0 a 255).

Fragmento de código

```
entrada = analogRead (1); //Asocia a la variable "entrada" el valor del pin 1
analogWrite (11, 128); //Direcciona al pin 11(es de tipo PWM) el valor de
tensión correspondiente a 128 (cercano a los 2,5 voltios)
```

8.5. FUNCIONES SERIAL

Las placas de Arduino poseen, en principio, un puerto serie para la comunicación con la computadora u otros dispositivos. Tal comunicación se produce en los pines TX y RX.

- La instrucción **Serial.begin()** inicializa el puerto serie y establece la velocidad de comunicación (especificada en baudios).

Fragmento de código

```
Serial.begin(9600); //abre el puerto serie y establece la velocidad de
comunicación en 9600bps
```

- La instrucción **Serial.println()** se usa para transmitir información por el puerto abierto. Al final les agrega carácter de retorno de carro (ASCII 13, o '\r') y un carácter de avance de línea (ASCII 10, o '\n'). De manera alternativa se puede usar la instrucción **Serial.print(),** la cual no agrega ningún carácter.

Fragmento de código

```
Serial.println(digitalRead(12)); //Envía el valor que hay en el pin 12
```

- **Serial.available()** devuelve el estado de la memoria intermedia (buffer) del puerto serie y así se puede chequear si hay datos recibidos dentro de este.
- **Serial.read()** lee la información que ingresa por el puerto serie.

Código: ejemploSerial

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if(Serial.available()>0) //comprueba si en el buffer hay datos
  {
 char dato=Serial.read();  //lee cada carácter,  almacenando en dato.
    Serial.println(dato);  //se imprime por consola lo recibido
  }
}
```

- También es posible enviar datos con la instrucción **Serial.write().** Esta escribe caracteres a través del puerto serie. Se pueden enviar una cantidad arbitraria de caracteres de una vez, a diferencia de Serial.read(), que lee de a uno por vez.

Código: ejemploSerial2

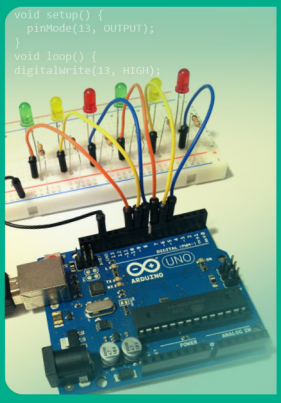
```
void setup()
{
  Serial.begin(9600);
}

 void loop()
{
  if(Serial.available()>0) //comprueba  si en el buffer hay datos
  {
    int dato=Serial.read();  //lee caracteres uno por uno

    Serial.write(dato);  //se imprime en consola el carácter recibido
  }
}
```

- La instrucción **Serial.flush()** permite vaciar de datos la memoria intermedia (buffer). Esto significa que si se llama a la funciones Serial.read() o Serial.available(), estas sólo devolverán los datos recibidos después de que se haya realizado la Serial.flush().
- Finalmente esta la instrucción **Serial.end(),** la cual desactiva la comunicación serie. Para activar la comunicación serie se debe llamar a la función **Serial.begin().**

Sobre los autores

FERNANDO RAÚL ALFREDO BORDIGNON es profesor asociado ordinario del Departamento de Tecnología en la Universidad Pedagógica de la Provincia de Buenos Aires (UNIPE), donde también dirige el Laboratorio de Investigación y Formación en Nuevas Tecnologías Informáticas Aplicadas a la Educación. Su formación inicial fue en ciencias de la computación, con especialización en redes de datos. También ha investigado temas relacionados con la comunicación y la educación. En la actualidad participa en proyectos de trabajo que analizan ambientes informales de aprendizaje como los *makerspaces*, los *hackerspaces*, o los *fab labs*, buscando experiencias que contribuyan a mejorar la educación pública.

ALEJANDRO ADRIÁN IGLESIAS es licenciado en Sistemas de Información por la Universidad Nacional de Luján. Se desempeña como jefe de trabajos prácticos del Departamento de Tecnología de la UNIPE e integra en esa misma universidad el equipo de investigadores del Laboratorio de Investigación y Formación en Nuevas Tecnologías Informáticas Aplicadas a la Educación. Participó en el desarrollo de videojuegos independientes y sus intereses están centrados en el uso de tecnologías de forma creativa, como herramienta de expresión y de empoderamiento ciudadano. Es autor del libro *¡Quiero hacer videojuegos!*, que UNIPE: Editorial Universitaria publicará en 2016.



ISBN 978-987-3805-12-7

9 789873 805127

unipe: editorial universitaria