Arduino en la Escuela: una herramienta versátil para la enseñanza de programación y robótica

Fernández, Gonzalo Pablo[1] gpfernandez@dc.uba.ar
Ticona Oquendo, María Belén[1] mticona@dc.uba.ar
Cossio-Mercado, Christian[1,2] ccossio@dc.uba.ar

[1] Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales. Departamento de Computación. Buenos Aires, Argentina.
[2] CONICET-Universidad de Buenos Aires. Instituto de Ciencias de la Computación (ICC). Buenos Aires, Argentina.

Resumen

En los últimos años cobró gran relevancia la enseñanza del Pensamiento Computacional, la Robótica y la Inteligencia Artificial en la educación de nivel primario y secundario. En este sentido, se han desarrollado varios entornos de programación por bloques para *Arduino*, los cuales se diferencian principalmente por el contexto educativo para el cual fueron pensados. En este trabajo presentamos *Arduino en la Escuela*, un entorno de programación basada en bloques para *Arduino*, creado para su utilización en ámbitos educativos. Se trata de un proyecto desarrollado de manera iterativa, fruto de la experiencia del uso de la herramienta en diversos espacios educativos: en la enseñanza primaria y secundaria, en formación y capacitación docente, y en divulgación científica en el nivel universitario, entre otros. Se caracteriza por su portabilidad, ya que no requiere conexión a Internet y tiene soporte para distintos sistemas operativos, por su facilidad de instalación y por haber sido desarrollado desde su origen en español. Todas estas características hacen de Arduino en la Escuela una herramienta útil para la enseñanza de programación, por ejemplo, en contextos de baja disponibilidad de recursos.

Palabras clave

Entorno de programación; Enseñanza de Programación; Arduino; Programación por bloques; Ciencias de la Computación

1. Introducción

Arduino es una familia de placas programables que ha adquirido gran popularidad tanto en entornos profesionales como educativos por ser simple, barata y de hardware abierto **[5]**. *Arduino en la Escuela* es un entorno de programación con fines educativos, específico para programar placas *Arduino*. Se caracteriza por usar programación por bloques, paradigma que ha demostrado facilitar la tarea de aprender a programar **[11, 12]**. Al abstraer detalles de implementación del mundo físico, es ideal para aprender los

conceptos básicos de la programación de este tipo de placas y, al no estar limitado por las especificaciones de la placa programada, permite explicar conceptos de electrónica aplicada generales. Adicionalmente, su alto nivel de personalización hace que sea adaptable tanto para un público escolar, docente, profesional y hasta para quien simplemente desea armar un proyecto personal en Arduino de forma sencilla y rápida. En la **Figura 1** se muestra la interfaz general de la herramienta.

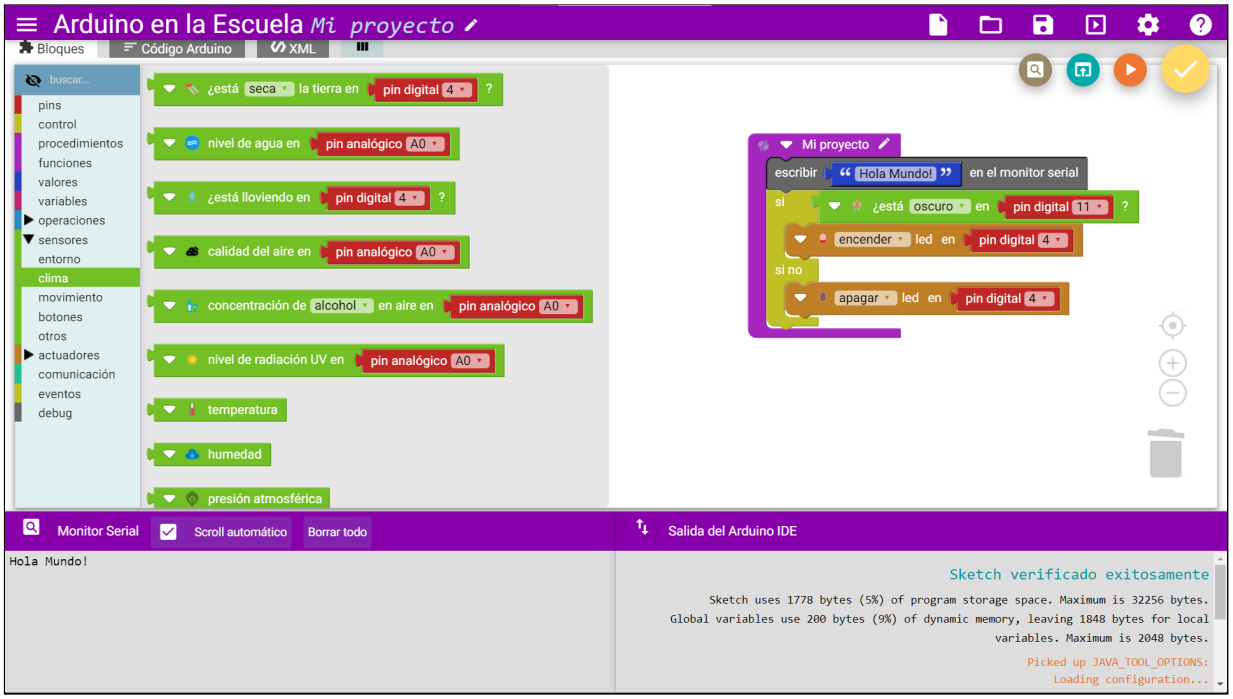


Figura 1: Interfaz gráfica de *Arduino en la Escuela*. Se muestra un programa de ejemplo (derecha), así como la caja de herramientas (izquierda), la salida del *IDE de Arduino* (abajo a la derecha) y la salida del monitor serial (abajo a la izquierda).

Las placas *Arduino* se usan cada vez más para enseñar conceptos de programación, electrónica y robótica. El lenguaje nativo para programarlas es *C++* que, a pesar de ser un lenguaje de alto nivel, presenta complicaciones a estudiantes sin experiencia, sobre todo frente a lenguajes como *Python*, cuya sintaxis es más similar a la del pseudocódigo y presenta mayor legibilidad. Varios estudios advierten acerca de la elevada curva de aprendizaje de *C++* y justifican de esta manera que se implementen entornos de programación por bloques, ya que presentan más similitudes con el lenguaje natural **[10]**.

Algunos de los entornos de programación de *Arduino* más recientes son *ArViz* **[1]**, *Ardestan* **[8]** y *BEESM* **[9]**. Sin embargo, ninguno presenta características destacables respecto a las herramientas en las que están basadas (*Ardublockly*[1] y *Blocklyduino*[2]), sino que son adaptaciones o extensiones de las mismas a contextos específicos.

[1] https://ardublockly.embeddedlog.com/index.html
[2] https://github.com/BlocklyDuino/BlocklyDuino/wiki

1.1 Contexto de la enseñanza de Programación y Robótica

El proyecto de *Arduino en la Escuela* se contextualiza en un panorama en el cual las Ciencias de la Computación se han instaurado como una disciplina esencial para el aprendizaje de habilidades lógicas y resolución de problemas. El razonamiento abstracto, el trabajo con problemas y el pensamiento creativo son capacidades que se desarrollan no solamente mediante el aprendizaje de las Matemáticas, sino también con la Computación. Si bien estas habilidades son comunes a ambas ciencias, dado que la forma tradicional de abordar la Matemática en la escuela es mediante problemas de abstracción numérica, las Ciencias de la Computación —que incluyen a la Robótica y a la Inteligencia Artificial, entre otras áreas— otorgan un enfoque diferencial que posibilita llevar la abstracción y resolución de problemas a un plano más general **[3]**.

Sin ir más lejos, el Pensamiento Computacional incluye capacidades como el reconocimiento de patrones, la descomposición de problemas complejos, la comprensión y el diseño de sistemas, la búsqueda de heurísticas para encontrar una solución, y el diseño de algoritmos, entre otros **[7, 13]**. Todas estas habilidades resultan importantes en la educación media para la comprensión de fenómenos y procesos complejos estudiados en ciencias, de forma de ayudar a comprender el mundo en el cual vivimos, además de su aprovechamiento para el seguimiento de estudios superiores.

Por todas estas razones la enseñanza de las Ciencias de la Computación se promueve en distintos niveles escolares para que forme parte de la currícula general y no solamente como conocimiento específico relacionado a la Informática. Su incorporación desde los niveles básicos de la escolarización está tomando gran importancia en todo el mundo, por lo que hay un interés generalizado en llevarlo a cabo de manera efectiva en el corto y mediano plazo. En particular, en Argentina hace varios años que se trabaja en cómo introducir a la programación y el Pensamiento Computacional como parte de la enseñanza obligatoria. La resolución N° 263/15 del Consejo Federal de Educación (CFE) de la Argentina remarca que es de "importancia estratégica para el sistema educativo argentino la enseñanza y el aprendizaje de la programación durante la escolaridad obligatoria, para fortalecer el desarrollo económico-social de la Nación"[3]. En este sentido, otra resolución más reciente del CFE, bajo número 343/18[4], aprobó los Núcleos de Aprendizaje Prioritarios (NAP) para Educación Digital, Programación y Robótica, estableciendo la obligatoriedad de la enseñanza de estos saberes en todos los niveles de la educación obligatoria. En esa misma resolución se había definido un plazo de 2 años para la adecuación de la currícula de las distintas jurisdicciones del país, de forma de cumplir con las pautas de los NAPs definidos.

No obstante, a pesar de la intención y su promoción institucional, aún no existen programas educativos, recursos preestablecidos ni metodologías estándares para llevar

[3] https://cfe.educacion.gob.ar/resoluciones/res15/263-15.pdf
[4] https://www.argentina.gob.ar/sites/default/files/res_cfe_343_18_0.pdf

las Ciencias de la Computación a todas las escuelas del país, ya que la implementación de la resolución del CFE citada es dispar, con resultados concretos recién en el nivel medio, y con baja cobertura en inicial y primario. Por otro lado, el campo de la Didáctica de la Computación se encuentra en una etapa de desarrollo inicial, donde queda mucho por investigar para conocer qué tipo de recursos son propicios a emplear de acuerdo a cada contexto educativo, lo que dificulta el cumplimiento de los objetivos planteados. Sin embargo, se espera un gran desarrollo del área en los próximos años, debido no sólo a las demandas a nivel gubernamental, sino también por los requerimientos de los sectores productivos, que requieren cada vez más personas formadas en programación y otros temas relacionados.

La Robótica viene con una rica historia en el campo de la investigación y, desde hace tiempo, también, con su utilización para la enseñanza de programación en diferentes niveles educativos. Esta elección radica en su naturaleza tangible la cual facilita la comprensión y el dimensionamiento del comportamiento lógico en estudio en un sentido físico, haciendo que su aprendizaje sea más atractivo e interactivo **[4]**.

El hecho de que se haya popularizado tanto la enseñanza de la Computación a través de la Robótica dio lugar al desarrollo de múltiples herramientas didácticas para facilitar la tarea de enseñanza, siendo los entornos de programación por bloques los predominantes **[6]**. Así, la gran variedad de entornos disponibles genera la duda de qué tanto se adecua cada uno a los objetivos didáctico-pedagógicos que definieron al momento de su creación, por lo que es necesario determinar qué entorno debería elegirse de acuerdo a cada contexto de uso. Esta clase de interrogantes resulta clave analizar ante la necesidad de incorporar las Ciencias de la Computación a la currícula escolar, en este caso, por medio de la robótica utilizando entornos de programación por bloques.

Debido a que estos entornos son los más usados para abordar la temática y dado el carácter disruptivo e innovador de la robótica como recurso pedagógico, en los últimos años gran parte de ellos se desarrollaron para trabajar con un hardware en particular, ya que en general forman parte de un proyecto que incluye el diseño de un robot específico. Si bien este enfoque es muy utilizado, tiene la desventaja de acotar el campo de aplicación de los conceptos aprendidos y, por lo tanto, limita los posibles contenidos a desarrollar a aquellos que se adaptan a lo permitido por el sistema. De esta forma se reduce la versatilidad de la herramienta utilizada, no permitiendo encarar problemas más generales ni profundizar su desarrollo de acuerdo a las necesidades de cada institución.

En contraposición a estos entornos dedicados, creados para un modelo particular de robot, existen otros más generales en los que no se programa un sistema particular sino que permiten escribir programas genéricos para una placa *Arduino*.

En este trabajo mencionaremos distintos aspectos de la herramienta *Arduino en la Escuela*, en tanto permitiría resolver los desafíos planteados anteriormente. Así, en la Sección 2 se eligen y se evalúan distintas características para los sistemas de

programación de placas *Arduino*. En la Sección 3 se hace una descripción de *Arduino en la Escuela*, mientras que en la Sección 4 se resumen algunas experiencias y posibles escenarios de uso para la herramienta. Por último, en la Sección 5 se realiza un análisis final de la herramienta, y se marcan posibles líneas de trabajo a futuro.

2. Revisión de herramientas de programación para *Arduino*

Resulta difícil determinar qué aspectos considerar al realizar una comparativa entre diferentes entornos de programación por bloques para *Arduino*. Las funcionalidades básicas son similares, pero difieren en ciertos aspectos importantes al momento de decidir cuál es el recurso más adecuado para el contexto deseado. Aunque muchas de las iniciativas apuntan a crear la herramienta más versátil, aplicable a cualquier contexto y a cualquier público, lo cierto es que cada una está diseñada con propósitos específicos, y con determinado modelo de *público objetivo* en mente, que no siempre es fácil de generalizar.

En este sentido, Melo y otros **[6]** afirman que no hay acuerdo en cuál es la mejor herramienta para enseñar a programar con *Arduino* debido a la necesidad de soportar múltiples contextos educativos. De esta forma, queda claro que no hay ni puede haber una herramienta que sea la más adecuada para cualquier contexto posible. Cada una de las existentes tiene ventajas y desventajas, así como situaciones en donde sería beneficioso usarla y otras en las cuales no es así.

2.1 Aspectos a analizar

A continuación resumimos cuatro aspectos seleccionados para la evaluación de las herramientas disponibles.

Facilidad de instalación

Para poder ser utilizado en las escuelas, el entorno debería instalarse en cada equipo que disponga cada institución. La tarea de instalar la aplicación en varios dispositivos puede ser extremadamente compleja y larga cuando los pasos a seguir para realizar la instalación no son lo más simples posibles, a lo que se le suma cualquier configuración adicional e instalación de dependencias que la aplicación pudiera requerir. Además, las computadoras en las escuelas suelen tener deshabilitados los permisos de administrador con lo cual una característica deseable es que se pueda usar como una aplicación ejecutable sin requerir instalación.

Multiplataforma

Si bien Windows sigue siendo el sistema operativo más común para usuarios finales, cada vez más se utilizan sistemas de código abierto, basados en Linux. En forma más

general, es importante el uso de software libre, más aún en espacios educativos de gestión estatal, en tanto permite el acceso irrestricto al conocimiento, y hace su aporte para alcanzar la soberanía tecnológica.

Así, las computadoras otorgadas por el Estado Argentino tienen ambos tipos de sistemas operativos y se espera que en los siguientes programas se distribuyan computadoras con *Huayra*[5], distribución de Linux desarrollada por el Estado en el marco del Programa Conectar Igualdad. Si bien esto no significa que necesariamente se deje de usar Windows, la tendencia es adoptar cada vez más los sistemas y programas de código abierto. De allí que sea deseable que los recursos para enseñar no se encuentren limitados a un sistema operativo en particular.

Independencia de internet

Muchas escuelas aún no tienen conexión a internet estable, y en algunas localidades tampoco tienen en los hogares. De esta manera, es importante garantizar la igualdad de condiciones para trabajar en el hogar ---donde también se espera que ocurra el acto educativo--- con las mismas herramientas usadas en el aula. Así, esperamos utilizar una herramienta en las escuelas que no dependa de la conexión a Internet para funcionar.

Disponibilidad en español

El uso del lenguaje nativo resulta primordial en los recursos didácticos a emplear en la etapa infantil, donde aún se encuentran en pleno desarrollo las habilidades de comunicación. La elección del vocabulario a emplear tanto para explicar conceptos así como también para que cada estudiante se apropie de saberes, debe hacerse considerando la riqueza expresiva del idioma. De allí que sea indispensable que el lenguaje usado por la herramienta y el que conozcan quienes la usan sea común, más aún si se quiere llevar una herramienta a cualquier escuela.

2.2 Análisis preliminar

Las herramientas más conocidas y usadas tienen como principal desventaja el idioma, ya que, si bien la mayoría tiene una versión en español, las traducciones suelen ser vagas o estar incompletas, y esto es problemático para alguien que está en la etapa inicial del aprendizaje de Robótica y Programación. A esto se le suman ciertas incoherencias en cuanto a términos utilizados en la propuesta didáctica que se quiere seguir, con respecto a los términos arbitrarios elegidos por la persona que realizó la traducción.

Como se mencionó anteriormente, es importante tener en cuenta la dependencia de la conexión a internet. La mayoría de las herramientas tienen versiones *offline*, pero algunas sólo funcionan en línea, lo que las descarta totalmente para su uso en escuelas sin

[5] https://huayra.educar.gob.ar/

acceso a internet. Además, al tener que ejecutarse desde un navegador, se complejiza técnicamente la forma en que la aplicación se conecta con la placa *Arduino*, ya que algunas de las herramientas exigen instalar un complemento para el navegador que no necesariamente funciona en cualquier sistema operativo. Las versiones *offline* también suelen tener la desventaja de ser específicas para determinadas plataformas, acotando también los espacios en los que pueden ser utilizadas.

Finalmente, se considera la disponibilidad de bloques específicos como un factor limitante tanto para la facilidad en el aprendizaje como para el abanico de posibles proyectos a encarar utilizando la herramienta en cuestión. Con muy pocos bloques se dificulta planificar y realizar proyectos medianamente interesantes, y la falta de bloques específicos puede hacer que ciertos proyectos sean demasiado complejos o, en ocasiones, imposibles. Tener demasiados bloques, en contraposición, dificulta el aprendizaje autoasistido y la exploración, ya que genera incertidumbre a la hora de elegir qué bloques usar. También hay que tener en cuenta qué tan expresivos son estos bloques: pocos bloques facilitan el aprendizaje, pero si estos son de muy bajo nivel tampoco se alcanza el efecto deseado.

Nombre	Fácil de instalar	Multiplataforma	Funciona sin internet	Está en español
TinkerCAD	Sí	Sí	No	Sí
SenseBlock	Sí	Sí	No	No
Scratch 4 Arduino	No	Sí	Sí	Sí
Minibloq	No	No	Sí	Sí
mBlock	No	No	Sí	En parte
Blocklyduino	Sí	Sí	Sí	No
Blockly@rduino	Sí	Sí	No	Sí
Blocklino	Sí	No	Sí	No
Arduino blocks	Sí	Sí	No	Sí
Ardublockly	Sí	Sí	Sí	En parte
Ardublock	Sí	Sí	Sí	No
Arduino en la Escuela	Sí	Sí	Sí	Sí

Tabla 1: Comparación de las principales características buscadas en la herramienta ideal para enseñar a programar *Arduino* en las escuelas. La columna “Fácil de instalar” hace referencia a si puede usarse sin necesidad de permisos de administrador (ya sea como aplicación ejecutable o como programa instalado en el sistema operativo). La columna “Multiplataforma” indica si además de funcionar en Windows, funciona en sistemas basados en Linux.

Los aspectos considerados son aquellos que consideramos relevantes para determinar qué herramienta utilizar. Como ya se ha mencionado, reflexionar sobre el contexto educativo en el cual se quiere aplicar la herramienta comparado con aquel para el que fue pensada resulta trascendental. A continuación profundizaremos en el entendimiento de las características del contexto para el cual fue pensado *Arduino en la Escuela*.

En la **Tabla 1** se presenta una comparación de las herramientas más conocidas de programación por bloques de *Arduino* a partir de las cuatro características mencionadas anteriormente. Como se observa, ninguna logra cumplir con todas satisfactoriamente.

2.3 Otras características evaluadas

En la **Tabla 1** se incluyen las principales características analizadas en este trabajo, pero también se incluyeron otras como parte del análisis.

Una de ellas es la **disponibilidad**. Si bien todas las herramientas analizadas son gratuitas, *Blockly@rduino*[6] requiere un *plugin* pago para poder bajar el código a la placa. Por otro lado, *TinkerCad*[7] y *Arduino blocks*[8] requieren registrarse para poder utilizar la herramienta.

También consideramos los **requerimientos técnicos de uso e instalación**. La mayoría de las herramientas analizadas tiene una versión *online*, con lo cual no sería necesaria instalación alguna, aunque a veces requieren de *plugins* adicionales para poder bajar el código a la placa. De aquellas que tienen una versión offline, *Ardublock*[9] necesita *Java* y *Blocklyduino*, *Blocklino*[10] y *Ardublockly* se pueden usar sin necesidad de instalar (i.e., no requieren permisos de administrador).

Al analizar la **documentación de usuario**, muy pocas tienen y sólo *Arduino blocks* tiene en español. Adicionalmente, algunas herramientas que formaron parte del análisis no tienen **mantenimiento reciente**, como es el caso de *Minibloq*[11], *Blocklyduino*, *Ardublockly* y *Ardublock*.

De todas las herramientas analizadas sólo *Blockly@rduino* y *Blocklino* permiten **personalizar el *toolbox***. Por otro lado, *Minibloq*, *Blockly@rduino*, *Blocklino*, *Arduino blocks* y *Ardublock* tienen acceso al **monitor serial incorporado**.

En cuanto a la **disponibilidad de bloques**, la mayoría cuenta con los básicos y más usados, aunque sólo *Blockly@rduino* y *Arduino blocks* proveen un extenso catálogo de bloques específicos. Con respecto a *Scratch 4 Arduino*[12], no se lo analiza como un entorno de propósito general para *Arduino*, sino que es sólo un intérprete de *Scratch* en la placa, por lo que no hay forma de ver el código *Arduino* generado por la herramienta.

---

[6] https://github.com/technologiescollege/Blockly-at-rduino
[7] https://www.tinkercad.com/dashboard
[8] http://www.arduinoblocks.com/
[9] http://blog.ardublock.com/
[10] https://github.com/fontainejp/blocklino
[11] http://blog.minibloq.org/
[12] http://s4a.cat/index_es.html

3 Características destacadas de Arduino en la Escuela

3.1 Funcionalidades

Un aspecto interesante que casi ninguna de las herramientas analizadas tiene en consideración es el control de ciertas restricciones en el código dependientes de la arquitectura de la placa. Un ejemplo muy simple es la asignación de roles para los pines. Al programar una placa *Arduino*, un pin puede ser configurado como de entrada o de salida, pero no ambos. Son muy pocas las herramientas que impiden utilizar un mismo *pin* para entrada y para salida, lo cual es muy importante a tener en cuenta ya que un usuario podría pensar que su programa es correcto sólo por el hecho de que el entorno de programación se lo permitió generar. Sin embargo, al intentar compilar el código obtendrá un error que probablemente no pueda interpretar.

Pensando en un público objetivo sin mucho conocimiento previo de programación, sería deseable que sea la herramienta y no el compilador quien pueda detectar estas fallas y ayude a quien la está usando con mensajes claros. Un buen ejemplo de esto es el sistema de advertencias de *Ardublockly*. Este sistema, heredado de *Blockly*, permite guiar a la persona que está usando la herramienta al momento de determinar por qué cierta combinación de bloques no es válida (véase **Figura 2.a**). Los mensajes claros son vitales para entender el problema y buscar cómo solucionarlo. *Arduino en la Escuela* utiliza el mismo sistema, pero, además, distingue entre advertencias y errores, de manera análoga a como lo hace la herramienta de programación de *Apps* de celular *AppInventor*[13].

Uno de los mecanismos que implementa *Blockly*[14] para facilitar el aprendizaje es impedir la conexión de bloques que visualmente parecieran encajar, para evitar un error de tipos (véase **Figura 2.b**). Esto se puede ver por ejemplo al intentar usar un bloque numérico como condición del bloque de alternativa condicional. Sin embargo, notamos que esto tiene algunas desventajas didácticas. Por ejemplo, a quienes están empezando a programar y no tienen ninguna capacitación respecto a teoría de tipos, les es difícil entender por qué un bloque numérico no puede usarse como condición del bloque de alternativa condicional. Durante el dictado de talleres por parte de nuestro equipo, se notó que los usuarios tratan de unirlos varias veces pensando que fue un error propio, hasta que se dan cuenta que es la herramienta la que no les deja hacerlo. Incluso después de haberse dado cuenta, no queda claro que entiendan la razón. Es por eso que *Arduino en la Escuela* en lugar de impedir que estos bloques se conecten, lo permite, y genera un mensaje de error indicando que la combinación es inválida (véase **Figura 2.c**).

Una de las grandes falencias en la mayoría de las herramientas es que al estar basadas en *Blockly*, no permiten hacer chequeo estático de tipos. *Blockly* fue desarrollado para lenguajes de tipado dinámico, con lo cual, no hay nada que impida, por ejemplo, asignar

[13] https://appinventor.mit.edu/
[14] https://developers.google.com/blockly

a una misma variable dos valores de tipos distintos. Esto es un problema al intentar generar código *Arduino*, ya que permite generar código que no se puede compilar. En cambio, *Arduino en la Escuela* implementa un sistema avanzado de chequeo de tipos[15] que le permite detectar muchas combinaciones inválidas, como, por ejemplo, asignación de una misma variable con valores de tipos distintos, argumentos de tipos inválidos en invocaciones a funciones o procedimientos, entre otros (véase **Figura 2.d**).

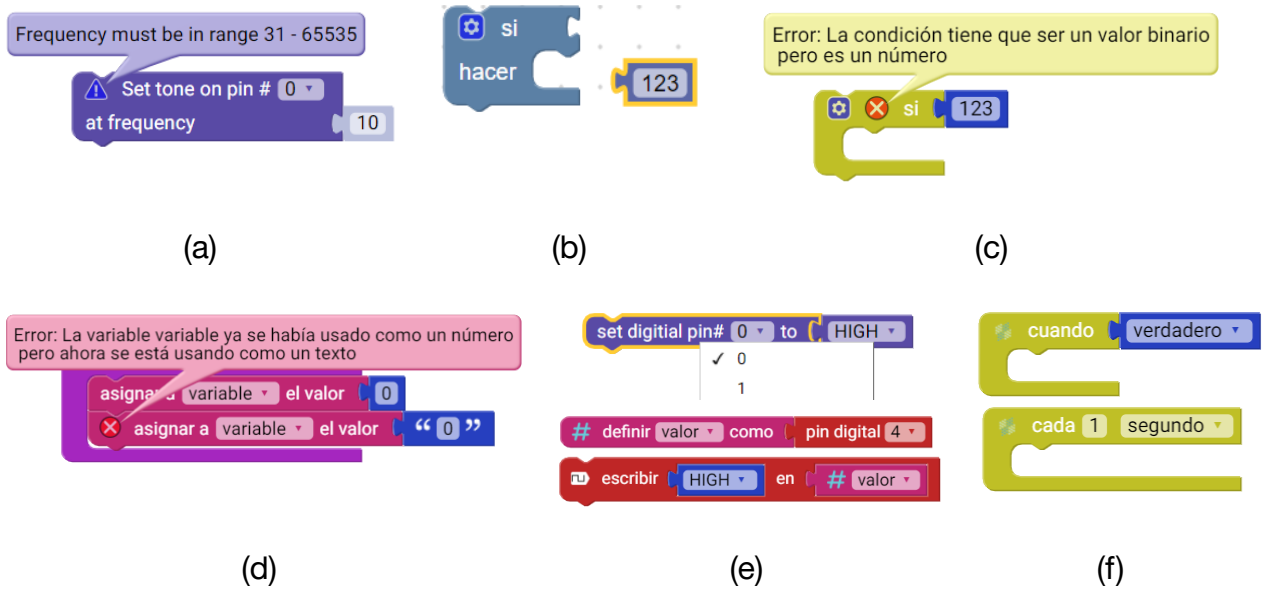


Figura 2: Funcionalidades a partir de los bloques que las implementan. (a) Advertencia de *Ardublockly* para indicar una combinación de valores inválidos. (b y c) Comparación entre *Blockly* (b) y *Arduino en la Escuela* (c) al intentar realizar una conexión inválida. (d) Detección de una asignación de variables inválida por error de tipado. (e) Comparación entre *Ardublockly* (arriba) y *Arduino en la Escuela* (abajo) en la forma de referirse a los pines de la placa. (f) Bloques ejemplos de *Arduino en la Escuela* para manejar eventos.

Otra funcionalidad relevante de *Arduino en la Escuela* es que permite tratar a los pines de la placa como un tipo más de datos. Es por esto que provee bloques específicos para los pines de forma de poder nombrarlos y reutilizarlos en varios lugares, a diferencia de la mayoría de las herramientas, en las que el pin debe seleccionarse de un menú desplegable (véase **Figura 2.e**).

Finalmente, se destaca el hecho de que Arduino en la Escuela permite trabajar en un modelo orientado a eventos gracias a su sistema de manejo de eventos, el cual se implementa como una capa de abstracción por encima de las interrupciones de hardware de *Arduino* (véase **Figura 2.f**).

[15] Se puede ver una demostración en https://gpfernandezflorio.github.io/blockly-inferencia/

3.2 Personalización

Arduino en la Escuela fue pensado para introducir a las personas a la programación y la electrónica aplicada. Con esos objetivos en mente, se desarrolló de forma que sea lo más simple posible y que abstraiga la mayor cantidad de información que pudiera entorpecer el aprendizaje. Sin embargo, también permite planificar proyectos más complejos habilitando configuraciones avanzadas. Un ejemplo de esto es la posibilidad de trabajar sólo con variables locales, sólo con variables globales, o permitir ambas. Como una opción por defecto en un contexto de aprendizaje inicial, sólo se pueda trabajar con variables locales, pero puede permitirse trabajar con variables globales con sólo unos clics.

Algo similar sucede con las advertencias que se muestran. Muchas veces estas son sólo sugerencias para ayudar en el proceso de aprendizaje, pero podrían ser molestas para alguien en una etapa más avanzada. Así, *Arduino en la Escuela* permite configurar qué tipo de advertencias se muestran y cuáles se tratan como errores, de la misma forma que lo hace un compilador.

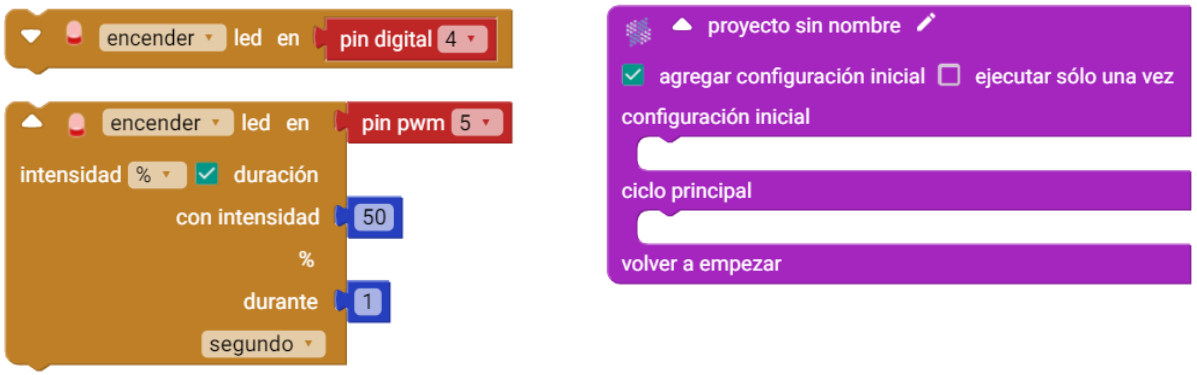
Otro elemento del sistema que se puede configurar es el conjunto de bloques disponible (o *toolbox*), ya que se puede elegir entre varios disponibles. Además, se permite crear otros personalizados, lo cual es especialmente útil para docentes que quieren planificar ejercicios específicos para los cuales necesitan sólo algunos bloques.

3.3 Usabilidad

Algunas características de *Arduino en la Escuela* permiten simplificar su uso y promover el aprendizaje. Uno de los diferenciales importantes es la incorporación del monitor serial a la interfaz web. Muchas de las herramientas existentes requieren abrir la interfaz del *IDE de Arduino* para poder acceder a esta información, mientras que en *Arduino en la escuela* todo forma parte de la misma aplicación. Adicionalmente, esto permite subir el código a la placa mientras el monitor serial está abierto, algo que siempre ocasiona problemas a quienes están empezando.

Algunas herramientas sólo proveen los bloques básicos de entrada y salida, lo que hace que sea difícil trabajar con proyectos grandes, mientras que otras tienen muchas variantes de los mismos bloques. Por ejemplo, un bloque para encender un led y otro para encenderlo durante una determinada cantidad de tiempo. Esto aumenta la capacidad expresiva del lenguaje, aunque tener demasiados bloques también puede ser un problema a la hora de encontrar el bloque adecuado para una determinada tarea. Algunas herramientas solucionan este inconveniente permitiendo personalizar el conjunto de bloques disponibles, y *Arduino en la Escuela* también posee esta característica, además de que permite encontrar bloques a través de un buscador, lo cual posibilita encontrar bloques específicos a partir del ingreso de texto. Esto también es muy útil para

aquellas personas que se inician en el tema, y aún no han tenido tiempo para familiarizarse con la interfaz o, más específicamente, con los bloques disponibles y su ubicación dentro del *toolbox*. Sumado a eso, implementa un sistema de opciones avanzadas para muchos de sus bloques ---principalmente los relacionados a sensores y actuadores--- que permite que un bloque sea lo suficientemente simple como para que sea fácil de entender, y que a la vez se pueda complejizar para obtener comportamientos más específicos (véase **Figura 3.a**). Otro ejemplo de esta propuesta es que el bloque principal también tiene un modo avanzado para configurar su funcionamiento, ya sea como un ciclo infinito o como una secuencia de instrucciones que se ejecuta una única vez (véase **Figura 3.b**). Entre estas opciones se encuentra también la posibilidad de implementar los procedimientos nativos de Arduino *setup* y *loop* para quien ya está en una etapa más avanzada del aprendizaje.



(a) (b)

Figura 3: Algunos de los bloques de *Arduino en la Escuela* que admiten personalización avanzada. Mediante la flecha blanca se expande el bloque y se muestran opciones avanzadas de configuración. (a) El bloque de led colapsado (arriba) y expandido (abajo). (b) El bloque principal expandido que permite agregar código para ejecutar en la inicialización del *Arduino* y definir si el programa principal se va a ejecutar una vez o repetirse indefinidamente.

Recientemente se le agregó a *Arduino en la Escuela* un nuevo modo que permite que se lo ejecute en un servidor web, para que esté disponible para accederse a través de internet. En este modo están disponibles todas las características excepto aquellas que requieren el *IDE de Arduino* para funcionar ---esto es, verificar el código, bajarlo a la placa y abrir el monitor serial---. Desde este modo se puede construir un programa y luego descargar el código *Arduino* generado para seguir editandolo en el editor de *Arduino*.

Finalmente se destaca la documentación de usuario. Aunque aún está en proceso de mejora, se está trabajando para que sea lo más descriptiva posible y que esté completamente en español. Un dato interesante es que desde la documentación se

pueden observar cada uno de los bloques disponibles y hasta interactuar con ellos. Posteriormente se espera agregar tutoriales y propuestas de ejercicios.

Respecto a la interfaz de usuario, algunas de las herramientas existentes están demasiado sobrecargadas, dificultando la tarea de organización para desarrollar la actividad. La gran disponibilidad de características de *Arduino* motiva a tenerlas todas accesibles a la vez, aunque esto podría abrumar a la persona que está comenzando a utilizar una herramienta. En este sentido, es preferible una interfaz simple, con pocas funcionalidades accesibles a simple vista. Las funcionalidades adicionales más específicas no deberían estar demasiado ocultas tampoco sino que deberían ser fáciles de encontrar en el momento que sean necesarias.

Muchos de los entornos analizados siguen este enfoque, y eligieron una interfaz simple y cómoda. Sin embargo, no siempre son fáciles de encontrar las características adicionales por lo que nunca llegan a utilizarse. Otro aspecto notable respecto a los entornos que prefieren este tipo de interfaz es que no suelen aprovechar el espacio de la pantalla, dejando mucho margen sin usar. *Arduino en la Escuela* se inclina por una interfaz simple, destinando la mayor parte del espacio visible en la pantalla inicial al sector de bloques. Todas las configuraciones y menús adicionales están sobre los bordes de la pantalla y pueden investigarse en el momento en que sean necesarios.

4. Algunas experiencias y posibles escenarios de uso de la herramienta

Por todas las características mencionadas en las secciones anteriores creemos que *Arduino en la Escuela* tiene un gran potencial para ser utilizado en múltiples contextos educativos. Puede utilizarse para enseñar conceptos generales de Programación y Ciencias de la Computación a personas que no tuvieron ninguna educación previa relacionada, ya que es ideal para introducir los conceptos a través de ejemplos del mundo real (“si está oscuro, apagar la luz”, “mientras no haya obstáculo, avanzar”) ignorando detalles sobre la electrónica. También puede ser utilizado para enseñar conceptos de electrónica aplicada, más allá de la programación, haciendo uso de los bloques de lectura de los sensores y su posterior observación en el monitor serial.

En particular, destacamos la experiencia de un taller dictado a principios de 2019, para estudiantes de 11 a 17 años, en el que se combinaron ambos enfoques para enseñar programación y electrónica aplicada a la vez. Incluso cuando la herramienta estaba aún en etapa de desarrollo, quienes participaron del taller expresaron su conformidad al utilizarla y apreciaron su versatilidad y facilidad de uso **[2]**. Durante el resto del año, mientras la herramienta seguía evolucionando, se continuó utilizando en distintos talleres, tanto para estudiantes como para docentes de distintos niveles educativos, actividades de divulgación y hasta en el curso docente *La Programación y su Didáctica*[16]. Sin

[16] https://www.dc.uba.ar/el-dc-apuesta-fuertemente-a-la-didactica-de-la-programacion/

embargo, *Arduino en la Escuela* no se limita a ser sólo una herramienta educativa, ya que la gran variedad de bloques y sus configuraciones avanzadas la convierten también en una excelente herramienta para encarar proyectos personales interesantes. Por ejemplo, en uno de los talleres realizados se lo utilizó para programar el funcionamiento de una estación meteorológica con estudiantes de escuelas secundarias, como parte de la actividad *Científic@s por un Día* de *Exactas-UBA*[17].

Otro escenario de uso podría ser como un paso intermedio para introducir a alguien que ya sabe programar y quiere aprender *Arduino* propiamente dicho, antes de pasar a programar en su *IDE*. Así, las dificultades que surgen de ir de un lenguaje a otro se pueden resolver de a poco, en lugar de tener que lidiar con todas juntas.

Finalmente, podemos pensar en una situación de alguien que no necesariamente quiere aprender a programar, aunque le gustaría poder desarrollar un proyecto usando *Arduino*. Todo esto hace de *Arduino en la Escuela* una herramienta altamente versátil.

5. Discusión y conclusiones

En este trabajo se resumieron las características principales del entorno de programación por bloques para placas *Arduino* llamado *Arduino en la Escuela*, que aunque guarda similitudes con otras herramientas ya existentes, posee características que lo hacen más adecuado para ciertos contextos. Nos concentramos en aquellas características que permiten utilizarla en la escuela, prestando particular atención a las necesidades de aquellas con baja disponibilidad de recursos. Las experiencias que hemos tenido con estudiantes han sido satisfactorias, con una buena recepción de parte de los usuarios. Cada taller, a su vez, sirvió para encontrar fallas y recibir sugerencias o pensar ideas sobre mejoras que luego fueron implementadas. De esta forma, concluimos que *Arduino en la Escuela* satisface los objetivos definidos al comienzo del proyecto.

Sin embargo, debemos analizar una limitación que surge como consecuencia de la virtualización de las clases debido a la pandemia. *Arduino en la Escuela* se planteó desde el principio como una herramienta para usarse en clases presenciales, de forma de aprovechar la riqueza de la tangibilidad de *Arduino* (es decir, que puedan interactuar físicamente con las placas). Si bien previamente al desarrollo observamos herramientas que proveen un simulador para probar el código generado sin necesidad de tener una placa física, como *TinkerCAD*, nunca lo consideramos una prioridad. La imposibilidad de probar el código sin tener una placa se ha vuelto una dificultad considerable en esquemas de clases virtuales, lo que queda como trabajo a futuro, de forma de incorporar un simulador de circuitos con *Arduino* para cubrir esta necesidad.

[17] dc.uba.ar/conociendo-y-midiendo-las-inundaciones-urbanas-mediante-el-uso-de-tecnologias/

Referencias bibliográficas

[1] Chochiang, K., Chaowanawatee, K., Silanon, K., & Kliangsuwan, T. (2019). *Arduino Visual Programming*. En *23rd International Computer Science and Engineering Conference (ICSEC)*.

[2] Fernandez-Florio, G., Cossio-Mercado, C., Macario-Cabral, D. & Reartes, C. (2019) *Electrónica aplicada para Ciencias y Tecnología con Arduino en la Escuela*. Póster presentado en *Segundas Jornadas Argentinas de Didáctica de la Programación*. Disponible en https://jadipro.unc.edu.ar/wp-content/blogs.dir/34/files/sites/34/2019/07/JADIPRO2019_paper_24.pdf

[3] Fundación Sadosky (2013). *CC-2016: Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas*. Disponible en http://www.fundacionsadosky.org.ar/wp-content/uploads/2014/06/cc-2016.pdf

[4] Horn, M.S., Solovey, E.T., Crouser, J.C., & Jacob, R.J.K. (2009). Comparing the Use of Tangible and Graphical Programming Languages for Informal Science Education. En *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, April 2009, pp. 975-984, doi: 10.1145/1518701.1518851

[5] Kushner, D. (2011). The making of Arduino. En *IEEE Spectrum, 26-11-2011*. Disponible en https://spectrum.ieee.org/the-making-of-arduino

[6] Melo, J., Fidelis, M., Alves, S., Freitas, U., & Dantas, R. (2020). A comprehensive review of Visual Programming Tools for Arduino. En *Proceedings of Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, doi: 10.1109/LARS/SBR/WRE51543.2020.9307023.

[7] Nardelli, E. (2019). Do we really need computational thinking?. En *Communications of the ACM*, Vol. 62, No. 2, pp. 32-35, doi: 10.1145/3231587

[8] Nishino, H. (2019). Ardestan: A Visual Programming Language for Arduino. En *The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology, pp 93-95, doi: 10.1145/3332167.3357126*

[9] Seraj, M., Autexier, S., & Janssen, J. (2018). BEESM, a Block-Based Educational Programming Tool for End Users. En *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*, pp. 886-891, doi: 10.1145/3240167.3240239

[10] Su, J., & Lin., T. (2018). Building a Simulated Blockly-Arduino-Based Programming Learning Tool: A Preliminary Study. En *7th International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 378-381, doi: 10.1109/IIAI-AAI.2018.00082.

[11] Weintrop, D., & Wilensky, U. (2015). To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-Based Programming. En *Proceedings of the 14th International Conference on Interaction Design and Children, pp. 199-208*, doi: 10.1145/2771839.2771860

[12] Weintrop, D. (2015). Minding the gap between blocks-based and text-based programming. En *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (2015), ACM, p. 720*, doi: 10.1145/2676723.2693622

[13] Wing, J. M. (2006). Computational thinking. En *Communications of the ACM*, Vol. 49, No. 3, March 2006, pp. 33-36, doi: 10.1145/1118178.1118215