Robótica en Fab Lab: Introducción a la programación para estudiantes de Ingeniería Mecánica

Robotics in Fab Lab: Introduction to programming for Mechanical Engineering students

Verónica Sofía D'Angelo [1], Guillermo Luján Rodríguez [1,2], María Florencia Sklate [2], Cesar Ignacio Pairetti [2] y Patricia Silvana San Martín [1]

[1] Instituto Rosario de Investigaciones en Ciencias de la Educación (IRICE: CONICET-UNR), Rosario, Argentina.
[2] Facultad de Ciencias Exactas, Ingeniería y Agrimensura (FCEIA: UNR), Rosario, Argentina.

dangelo@irice-conicet.gov.ar

Resumen. En este trabajo se analiza una experiencia piloto de la actividad "Taller de robótica" en el Fab Lab UNR, un laboratorio *spacemaker* creado en 2020 por la Facultad de Ciencias Exactas, Ingeniería y Agrimensura de la Universidad Nacional de Rosario, en el marco de las políticas de fortalecimiento de la educación de ingenieros. El equipo de investigación interdisciplinar indagó sobre los supuestos pedagógicos apropiados para introducir conceptos básicos de programación a estudiantes de ingeniería mecánica, con especial énfasis en: trabajo colaborativo, aprendizaje basado en problemas, motivación inicial, articulación horizontal entre asignaturas y balance adecuado entre conceptualización y procedimientos. Los primeros resultados muestran que la percepción de los estudiantes es positiva y que la secuencia didáctica resultó apropiada para trabajar con ingresantes, lo que confirma el potencial de estas prácticas de taller para aprender programación en el ciclo introductorio de carreras de ingeniería.

Palabras claves: Introducción a la Ingeniería, Programación, Robótica, Fab Lab, Espacios *Maker*, Ingeniería Mecánica, Modalidad taller, Arduino.

Abstract. In this work, a pilot experience of the "Robotics Workshop" activity at Fab Lab UNR is analyzed. It is a *spacemaker* laboratory created in 2020 by the Faculty of Exact Sciences, Engineering, and Surveying of the National University of Rosario, within the framework of the policies to strengthen engineering education. The interdisciplinary research team inquired about the appropriate pedagogical assumptions to introduce basic programming concepts to mechanical engineering students, with special emphasis on collaborative work, problem-based learning, initial motivation, horizontal articulation between subjects, and an adequate balance between conceptualization and procedures. . The first results show that the perception of the students is positive and that the didactic sequence

Received January 2023 Accepted April 2023; Published May 2023
https://doi.org/10.24215/15146774e007

was appropriate to work with newcomers, which confirms the potential of these workshop practices to learn to program in the introductory cycle of engineering careers.

Keywords: Introduction to Engineering, Programming, Robotics, Fab Lab, *Spacemakers*, Mechanical Engineering, Workshop modality, Arduino.

1 Introducción y marco general

Este trabajo presenta un estudio exploratorio en torno a la actividad "Taller de robótica" integrada en la asignatura "Introducción a la Ingeniería Mecánica" en la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (FCEIA) de la Universidad Nacional de Rosario (UNR), Argentina. En el marco de las políticas educativas para el fortalecimiento de la formación de ingenieros [1]–[3], la carrera de Ingeniería Mecánica en el Plan de Estudios 2014 incorporó la modalidad pedagógica de taller físico-virtual en esta actividad curricular introductoria, para realizar prácticas relacionadas con contenidos de primer año [4]. Desde 2020, estas actividades se desarrollan en Fab Lab UNR [5], un laboratorio construido especialmente para realizar proyectos de diseño, prototipado y manufactura. Este laboratorio, ubicado en la Escuela de Ingeniería Mecánica (EIM) del Centro Universitario Rosario, cuenta con fresadora y torno CNC, impresora 3D, pantógrafo láser, y diversas máquinas pequeñas, herramientas de mano e instrumentos de medición. Se cuenta con una gran variedad de equipamiento de electrónica vinculado a la robótica: placas Arduino de diversas características, motores paso a paso, bombas de impulsión, sensores diversos, etc. La actividad "Taller de Robótica", se brinda en paralelo con otras actividades, tales como Impresión 3D y Mecanizado CNC.

Las actividades docentes del Fab Lab UNR se desarrollan en colaboración con el Instituto Rosario de Investigaciones en Ciencias de la Educación (IRICE: CONICET-UNR) a través del equipo de Dispositivos Intermediales Dinámicos [6]. Esto hace de Fab Lab UNR un entorno de trabajo particular, ya que participan investigadores con base en ciencias sociales, que contribuyeron a establecer ciertas premisas pedagógicas para el diseño de actividades.

Dichas premisas provienen de un marco teórico multidimensional que aporta puntos de vista diferenciados sobre el objeto de estudio, entre los cuales destacamos el punto de vista institucional, el social, el tecnológico y el educativo. Dentro de este último se conciben las Prácticas Educativas Mediatizadas (PEM) de corte constructivista (Andrés & San Martín, 2019). El constructivismo es la base de teorías y enfoques que conciben al sujeto como productor de su propio conocimiento dando origen a las denominadas metodologías activas, dentro de las cuales, algunas se orientan hacia los enfoques situados y otras buscan basamento en la psicología del aprendizaje. Dentro de las primeras se encuentran los enfoques socioculturales [8], y el aprendizaje basado en problemas/proyectos [9] [10]. Entre las segundas, son de especial interés en esta investigación las teorías de Bruner sobre la secuenciación de actividades [11] y su organización del currículum en espiral [12], el trabajo posterior de Ausubel sobre los organizadores previos [13], y la teoría de la elaboración que retoma los trabajos de

Ausubel y Bruner y los complementa con las posteriores investigaciones sobre memoria y aprendizaje [14].

1.1 La perspectiva psicológica: importancia de la diferencia entre conocimiento procedimental y declarativo

El surgimiento de los Fab Lab nos obliga a re examinar las teorías del aprendizaje en torno a la distinción entre conocimiento declarativo y procedimental.

Tradicionalmente, la educación se ha ocupado de la *transmisión* de conocimientos *declarativos* (teóricos, que se pueden expresar verbalmente) en oposición a los conocimientos *procedimentales* (resolución de problemas, habilidades, generalmente no conscientes ni explicitables). Este último tipo de conocimiento merece especial atención por ser uno de los componentes de una "competencia" [15], pero sobre todo, porque repentinamente adquirió relevancia con las iniciativas de pensamiento computacional y del movimiento *maker*, y porque tradicionalmente ha sido el tipo de conocimiento menos explorado en el ámbito educativo. Incluso en la época en que Ausubel desarrolló su teoría del aprendizaje significativo e instaló la premisa básica de que todo aprendizaje debe estar anclado en saberes previos, no se conocía la distinción entre conocimientos declarativos y procedimentales.

Un problema común en el aprendizaje del conocimiento procedimental es la interpretación errónea de que el aprendizaje *activo* consiste simplemente en "hacer algo". Es decir, que se aprende por el mero hecho de "hacer", sin participación del conocimiento declarativo. Por el contrario, el conocimiento declarativo es necesario en la fase inicial del aprendizaje de procedimientos. Las personas recuerdan las experiencias "en función de las actividades *mentales* que realizan en el curso de esa experiencia" [16]. No todos los tipos de práctica son igualmente eficaces para formar expertos en ingeniería. Mientras que la práctica repetitiva permite formar expertos en procedimientos rutinarios (obreros o técnicos), la práctica reflexiva que incluye conceptos apunta a formar expertos en estrategias de resolución de problemas (ingenieras e ingenieros) [17]. Lo que caracteriza a un experto no es la cantidad de práctica o la velocidad de respuesta sino su capacidad estratégica (para la toma de decisiones guiada por conocimiento conceptual).

En síntesis, el aprendizaje de procedimientos se puede realizar de manera significativa o no significativa. En el primer caso, durante experiencias reflexivas que habilitan la conceptualización. En el segundo caso, de manera pasiva y repetitiva sin participación de la creatividad ni la comprensión. Cuando la participación del conocimiento declarativo es débil, ya sea porque se trata de un hábito adquirido informalmente de modo precipitado, o porque en la clase no se definió adecuadamente el concepto subyacente, o bien porque los estudiantes no están motivados para comprender las explicaciones sino apresurados por realizar la práctica como fin en sí misma, se suele observar que algunos estudiantes realizan la tarea pero *sin comprender* conceptualmente *qué es* lo que están realizando" (conocimiento declarativo).

Si bien las contribuciones de Bruner [11] [12] y Ausubel [13] fueron pioneras en hallar un fundamento en la psicología del aprendizaje y la memoria de sus diseños pedagógicos, estas teorías sólo eran aplicables al conocimiento declarativo: los mapas

conceptuales sirven para conectar conceptos pero *no para organizar procedimientos*. Sólo a la luz de investigaciones posteriores sobre la memoria se pudo advertir la diferencia entre el aprendizaje de conocimiento declarativo y procedimental, y la importancia de la *elaboración* en la adquisición del conocimiento.

La teoría de la elaboración [14] retoma el trabajo de Bruner y Ausubel y los nuevos hallazgos de las investigaciones sobre memoria [16] para aportar sugerencias sobre cómo deben planificarse los contenidos respetando la naturaleza del conocimiento, sea este declarativo o procedimental. Como su nombre lo indica, "elaborar", cuando se trata de una actividad práctica, no es simplemente *hacer* sino *hacer pensando en lo que se hace*. Las secuencias de actividades sugeridas por Reigeluth no son lineales, se basan en siete principios: (1) secuencia de elaboración (iniciada por un epítome), (2) conocimientos previos necesarios, (3) resumen, (4) síntesis, (5) analogía, (6) estrategias cognitivas y (7) control del alumno. Por ejemplo, el primero de estos principios, apunta a evitar la fragmentación del saber. Reigeluth observa que los docentes, al realizar sus planificaciones, dividen los temas de manera secuencial, a efectos de organizar el tiempo, y esta fragmentación arbitraria, repercute negativamente en la motivación de los estudiantes. En cambio, sugiere comenzar la planificación con una primera actividad que denomina "epítome", que condensa la totalidad de lo que se pretende enseñar durante un curso (materia) en un único evento (clase, encuentro, exposición). Esa primera clase o encuentro debe presentar los elementos esenciales de modo reducido, en pequeña escala. Esto es coherente con limitaciones de la memoria humana en cuanto a la cantidad de ítems que pueden atenderse simultáneamente y a la necesidad de elaborar repetidamente un concepto para fijarlo a largo plazo. A medida que se repiten los encuentros, o dentro de una misma clase, se debe volver sobre el mismo concepto, cada vez con mayor profundidad. De este modo, toda secuencia debe partir desde una totalidad simple hacia la profundización de cada elemento, es decir, de lo simple a lo complejo (o de lo general a lo particular: la equivalencia procede de que lo general, al no contener especificaciones detalladas, es más simple que lo particular). Esta macro teoría, por ser precursora en el abordaje pedagógico del aprendizaje procedimental complejo, está siendo adoptada por pedagogos del pensamiento computacional [18] y del movimiento maker en Fab Labs [19], ya que el aprendizaje de la programación es considerado un ejemplo paradigmático de aprendizaje complejo [20] [21].

Los encuentros que se planificaron en Fab Lab UNR guardaban expectativas de esta índole: por un lado, poder presentar a los estudiantes una primer "fotografía" del todo, un encuentro que, en pequeña escala, anticipe los encuentros futuros con las asignaturas que ampliarían la temática, una promesa de aprendizaje significativo, que sólo se cumple cuando la teoría puede mostrar su sentido en una práctica coherente, y un deseo de aliviar la carga extremadamente teórica que constituye el recibimiento tradicional a los ingresantes de ingeniería.

1.2 La perspectiva desde las ciencias humanas y sociales

Desde un marco socio-técnico-cultural no determinista, no sólo se asume que la tecnología puede dar forma a la sociedad sino también la sociedad a la tecnología [22]

[23]. De este modo, los dispositivos tecnológicos no se conciben como implementaciones cerradas que se replican de manera idéntica en todos los contextos, sino como un saber maleable que también "se deja" modelar por dichos contextos. Esta dialéctica cobra especial relevancia en el análisis de procesos de aprendizaje mediados por tecnología en los que se busca la igualdad de oportunidades.

Una consecuencia esperada de la emergencia de la cultura *DIY* (*Do It Yourself*) es la democratización del conocimiento y de la invención. Sin embargo, la equidad no está garantizada por el acceso físico a los recursos tecnológicos, como puede observarse en las investigaciones sobre brecha digital [24]–[26], sino también por el desarrollo de los procesos superiores de pensamiento [27] [28], que permitan a los estudiantes la apropiación. Esto es, se garantiza la apropiación cuando se asegura un marco social, institucional, tecnológico y educativo equitativo.

No todos los aprendices son intelectualmente "activos" en su autogestión del aprendizaje. Se ha observado, por ejemplo, que algunos estudiantes sólo son "espectadores" de los cursos en video, y en estos casos, se les recomienda realizar actividades durante la observación [29]. Otro comportamiento pasivo es la realización de actividades pero sin reflexionar sobre ellas, por ejemplo, cuando se sigue un video paso a paso imitando cada uno de los pasos para construir un objeto pero sin prestar atención a lo que se está haciendo (sin desarrollar el concepto). La preocupación por la "reflexión" como parte de las actividades *maker* es un tema recurrente en las investigaciones de la comunidad *FabLearn* [30].

Fuera de los espacios institucionales, los videos que prometen hacer hasta lo más complejo en pocos minutos, promueven una cultura *maker* para la creación de artefactos sin intervención de la comprensión ni la creatividad. Es por ello que interesa distinguir entre la "cultura *maker*" como movimiento global de aprendizaje informal y los "espacios *maker*" que han tenido lugar a nivel internacional en los ambientes educativos formales.

Los *Fab Lab* son laboratorios de fabricación en contextos educativos [31] caracterizados por ciertas particularidades. Por un lado, los talleres se desarrollan en paralelo con la emergencia de la comunidad *FabLearn* conformada por los investigadores y educadores a nivel internacional que estudian la evolución de los Fab Lab con el objetivo de mejorar las prácticas [32]–[35].

Las raíces teóricas de estos espacios *maker* se remontan al constructivismo [36] [37], al construccionismo de Papert [38], a la pedagogía crítica [39] [40] y al aprendizaje basado en proyectos [41]. Pero se distinguen de los talleres tradicionales en que utilizan tecnologías de reciente aparición -como la impresión 3D, el maquetado láser, las placas Arduino en robótica- que confluyen con plataformas de software que facilitaron el acercamiento del estudiante al diseño y la programación, tales como el lenguaje Logo [42], sus sucesores Scratch [43] [44], y NetLogo [45], y los lenguajes desarrollados para placas Arduino.

En los contextos educativos, se espera que los Fab Lab estén abocados a transmitir conceptos (apoyados en una práctica concreta), con la guía de profesionales expertos en cada disciplina y en relación con otras asignaturas de la carrera, sin perder su carácter de taller.

La idiosincrasia y los objetivos de cada carrera se complementan de modos diversos con la iniciativa del pensamiento computacional. No es lo mismo aprender a programar con el objetivo en mente de transformarse en un diseñador de videojuegos, que aprender a programar para componer música o aprender a programar para ser ingeniero.

Cada año, los alumnos ingresantes a la carrera de Ingeniería Mecánica UNR escriben sus biografías en una actividad solicitada por el equipo de profesores de Introducción a la Ingeniería Mecánica. En estas biografías vuelcan sus intereses, inquietudes, la historia de por qué han elegido esa carrera. Si bien el imaginario del ingresante a ingeniería mecánica suele incluir expectativas poco realistas sobre las posibilidades de la profesión, es claro que éstas giran en torno a la manipulación de componentes físicos y dispositivos móviles. Si bien se realizó una experiencia previa de programación con lenguajes visuales [46], el equipo llegó a la conclusión de que la programación con placas Arduino de un robot con funcionalidad simple, es más apropiada para los estudiantes de ingeniería mecánica, por una mejor adecuación a sus expectativas.

1.3 El profesor acompaña a construir el *concepto*, no sólo el objeto

Como se mencionó anteriormente, la preocupación de los profesores por dar lugar a la reflexión dentro de las prácticas maker aparece en numerosas investigaciones sobre espacios *maker*. Sin embargo, no hay un consenso claro acerca de qué procesos específicos intervienen en la reflexión [30].

Desde los inicios de la iniciativa para la promoción del pensamiento computacional se estableció claramente que no se trataba de transmitir únicamente habilidades sino conceptos [49] [50]. Ambos aspectos son componentes de una competencia [15]. Esta afirmación que parece estar clara a nivel teórico, en la práctica *maker*, donde la balanza se inclina hacia lo procedimental, se está produciendo lentamente una relegación de lo conceptual a un segundo plano.

El desafío en el diseño de esta consigna, fue decidir en qué etapa de la actividad introducir los conceptos sin obstaculizar el procedimiento. Esto se logró problematizando la actividad, generando preguntas sobre cómo mejorar o corregir los problemas que se planteaban y volviendo sobre los problemas para re elaborarlos iterativamente en una secuencia espiralada.

Para explicar la relación entre conocimiento declarativo y procedural, utilizaremos como analogía el proceso de aprender a andar en bicicleta. Cuando una persona sabe andar en bicicleta no necesita prestar atención a los movimientos de sus pies sobre los pedales. De hecho, si le pidiéramos que tome conciencia de cómo va pedaleando podría incluso caerse por observar sus pies para explicitar algo ya inconsciente. Sin embargo, en su etapa de aprendizaje en la niñez probablemente habrá seguido instrucciones y habrá pensado detenidamente dónde colocar los pies y cómo mantener el equilibrio. Los consejos recibidos en ese momento suelen ser bienvenidos y captados con atención. Se dice que el conocimiento se encuentra en un estadío *declarativo* [51]. A medida que se repite la misma práctica, el conocimiento se vuelve procedural y automático, adquiere una especie de inercia. Una vez automatizado el saber andar en bicicleta, es difícil volver a hacer explícitos los primeros pedaleos.

La analogía permite vislumbrar que cuando los aprendices de videos expeditivos ya han automatizado la práctica de ensamblar, subir programas desde Internet a un robot y verlo funcionar sin necesidad de implicar su propio razonamiento lógico, probablemente se resistan a volver sobre los primeros pasos para cuestionarse: cómo está escrito el código, para qué sirve cada instrucción, o cuál es el concepto de variable. Cuando surjan dificultades, se van a tornar irresolubles por desconocer el origen de los errores, y es probable que pierdan la motivación y abandonen la tarea. Adquirir un automatismo no garantiza que el aprendizaje adquirido sea correcto. Todo aprendizaje procedural conlleva el riesgo de automatizar algo equivocado. Una buena oportunidad que tienen los profesores es la de introducir conceptos durante actividades prácticas, es decir, *antes de la automatización*, cuando la actividad está en su *etapa declarativa*, por tanto, en los primeros pasos del aprendizaje.

El conocimiento declarativo es verbal. Se expresa en palabras. Esto también es particularmente importante en un ambiente *maker* porque debe evitarse la posibilidad (también existente) de trabajar *sin hablar*. Un modo de activar la explicitación verbal es intentar resolver problemas en equipo, otro modo es que el docente explicite claramente el nombre de cada término y su definición, y que plantee preguntas. Investigaciones recientes sobre aprendizaje de conceptos en diseños instruccionales revelan que una forma efectiva de enseñar un concepto es mostrar su definición de manera contigua a los ejemplos [52]–[54]. En contraste con la típica separación entre teoría y práctica, el componente teórico del concepto (su definición) no está separado del componente concreto (el ejemplo).

En vez de pensar en el taller como un lugar para poner en práctica la teoría del aula, sería más apropiado organizar la práctica en el taller de modo que las definiciones se den durante el procedimiento en el momento preciso en que surge el ejemplo. Por ejemplo, en la experiencia *maker* de robótica, cuando se desea encender un motor, debe cambiarse el valor de la variable. Es un buen momento para presentar el concepto de variable informática[1] con una definición breve: “un espacio de memoria en el que se almacena un valor que puede ser leído o modificado”.

1.4 Motivación

Otro problema que surge al diseñar una actividad breve con un objetivo complejo es cómo mantener la motivación en la tarea. Según Deci y Ryan [55], la motivación intrínseca, está basada en tres necesidades psicológicas: (1) curiosidad, (2) percepción de causación personal (de ser el autor del resultado) y (3) percepción de efectividad (que el resultado sea adecuado). No es sencillo lograr satisfacer cada una de estas necesidades durante actividades de enseñanza porque la relación entre ellas puede dar lugar a un desequilibrio.

Por ejemplo, la percepción de causación personal se puede lograr permitiendo que los alumnos trabajen solos. Pero la percepción de efectividad (3) sólo se obtiene cuando el resultado es correcto: el artefacto construido funciona. Para evitar que los alumnos

[1] Se aclara “variable informática” para distinguirlo del concepto de variable matemática, ya conocido por los alumnos.

se frustren por no lograr hacer funcionar el dispositivo, interviene el profesor, pero su intervención no debe ser tan abrumadora que el estudiante no perciba su propia causación personal y sienta que lo construyó el profesor.

Por otra parte, para que algo despierte curiosidad, no debe ser totalmente conocido ni totalmente desconocido. Un criterio para que la actividad no fuera totalmente conocida fue proponer instrucciones no típicas, es decir, que no pudieran hallarse en Internet fácilmente. Para que la actividad fuera parcialmente conocida (lo que también estimula la participación) se propuso a los alumnos que eligiesen ellos mismos en qué actividades de Fab Lab inscribirse. Evidentemente, las que más les interesen deben ser conocidas parcialmente, y la posibilidad de elegir también impacta positivamente sobre la sensación de causación personal.

1.5 Contenidos

Al equilibrio entre aspectos cognitivos y motivacionales se sumaba el requisito pedagógico de transmitir contenidos durante la práctica. El objetivo principal era proponer conceptos de programación a estudiantes de ingeniería mecánica vinculados al área específica, pero con miras a una recuperación futura de dichos contenidos en la asignatura Informática. Los objetivos secundarios eran: (a) lograr un primer acercamiento con los conceptos de robótica y de programación, (b) reflexionar sobre temas trascendentes para el desarrollo de la ingeniería mecánica como la predicción de eventos en un contexto controlado, (c) experimentar con un proceso de ajuste de variables, (d) reflexionar sobre la precisión de las mediciones indirectas, (e) aplicar procesos de validación e iteraciones de mejora, (f) aprender en un entorno colaborativo.

Para lograr introducir estos conceptos en la fase declarativa del aprendizaje de la construcción de un robot, se debía planificar anticipadamente en qué momento de la actividad se introduciría cada uno, aprovechando las cuestiones que surgirían durante la programación del robot.

2 Método

El abordaje metodológico fue de tipo exploratorio, con características de investigación acción, ya que se intentó producir conocimiento durante el desarrollo de la actividad "Taller de Robótica". Los encuentros se repitieron semanalmente con diferentes grupos de alumnos. Se analizó la experiencia de cada encuentro y se realizaron ajustes basados en ese análisis.

El equipo de investigación estuvo integrado por un doctor en Ingeniería, una doctoranda en Ingeniería -ambos docentes de la asignatura Introducción a la Ingeniería Mecánica-, y una doctoranda en Psicología, que observaron el desarrollo de la actividad y analizaron los datos recabados. Por un lado, se registraron datos cualitativos: ambos docentes escribieron bitácoras semanales donde volcaron sus percepciones sobre el encuentro: cuestiones generales, comentarios de alumnos, observaciones que pudieran ser útiles para la reflexión desde el punto de vista de la didáctica de la ingeniería

mecánica. La doctoranda en Psicología realizó un registro similar de observaciones desde el punto de vista técnico y pedagógico.

Además de las observaciones y bitácoras, los estudiantes completaron una encuesta. Las variables indagadas en la encuesta fueron la percepción de los estudiantes de los diversos aspectos de la experiencia, como la claridad en la consigna y la metodología de trabajo, la adecuación del espacio físico y de las herramientas e instrumentos. Se indagó a su vez, cuáles fueron los contenidos de otras asignaturas que los estudiantes pudieron integrar en la experiencia, los criterios funcionales utilizados en la resolución del problema, la relación con otros estudiantes y con los docentes, y las sugerencias para próximas actividades.

3 Experiencias y observaciones

Se utilizó el robot educativo Múltiplo N6 diseñado en Argentina por RobotGroup [56] (Figura 1) que consta de dos motores independientes de corriente continua para dos ruedas delanteras, una rueda trasera más pequeña con movimiento libre, dos sensores infrarrojos reflexivos para medir luz y un sensor ultrasónico para medir distancia.

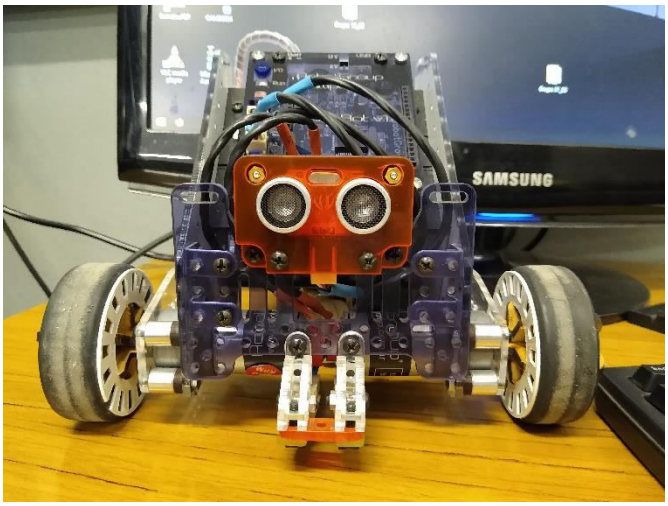

Fig. 1. Robot Multiplo N6 de RobotGroup ensamblado.

El kit contiene un controlador DuinoBot 2.4 compatible con Arduino programable con Arduino Ide o con Minibloq, se utilizó éste último por poseer interfaz gráfica y posibilidad de observar el código generado a la vez [57].

La actividad tenía una duración de 3 horas. Se reprodujo en varias oportunidades con diferentes grupos que no superaron los seis participantes hasta cubrir un total de 51 estudiantes que se inscribieron libremente para participar. Las opciones eran Robótica, Impresión 3D, Maquinado CNC, Estructuras estáticas y Electro neumática.

La sala de trabajo contaba con un pizarrón móvil, una mesa con una disposición que permitía la conversación de frente, una segunda mesa con una computadora y un banco alargado frente a ella con capacidad para tres alumnos y un espacio para que otros estuvieran de pie. Se contaba con un espacio libre de 15 metros cuadrados para la prueba de funcionamiento del robot. El desplazamiento por los espacios era funcional a las tareas.

3.1 Secuencia didáctica espiralada

Las cuatro consignas de programación eran básicas y se ordenaban desde la más simple a la más compleja. La consigna se comentaba primero en la mesa de charla. Cuando una idea surgida era importante, se anotaba en el pizarrón. Cada consigna se problematizaba oralmente en equipo, luego se escribía el *pseudo código* en el pizarrón y sólo cuando era evidente que los estudiantes habían comprendido la lógica de las instrucciones, se pasaba a la computadora para programar en Minibloq (Figura 2).

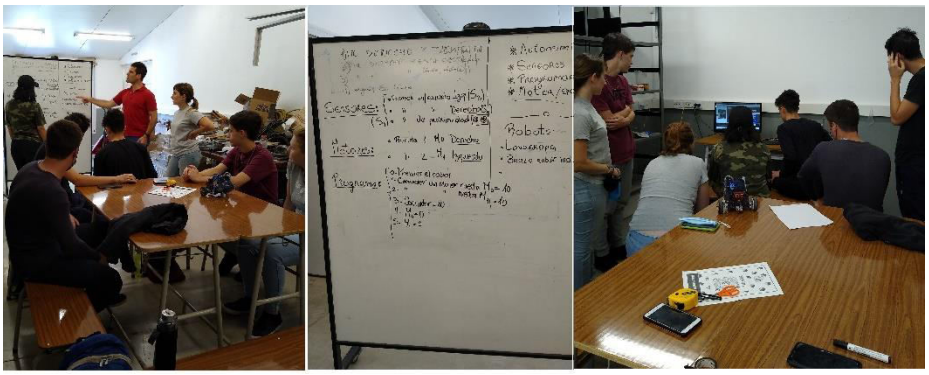

Fig. 2. Secuencia de producción previa a la prueba del robot: (1) mesa de charla, (2) pizarrón de conceptos, (3) programación en Minibloq.

Algunos participantes escribieron código, otros colaboraron aportando ideas sobre cada instrucción, pero este orden era flexible, porque el grupo estaba en constante movimiento, una vez finalizado el programa, se subía el código al robot, se trasladaba a la zona libre para probarlo (Figura 3) y el equipo se reorganizaba en la siguiente tarea de programación. Se trató de evitar que las mismas personas programaran todas las consignas.

Fig. 3. Zona de prueba del robot luego de la programación.

Durante las pruebas se analizaban las diferencias entre lo que se había planificado (e imaginado) y el resultado real de ese funcionamiento para actuar en consecuencia (indagando las discrepancias o proponiendo modificaciones).

Los objetivos prácticos en cada una de las consignas en la columna "Descripción", en la fase de "Práctica" (ver fila 3 en Tabla 1) buscan transmitir nociones básicas de mecánica entrelazadas con los conceptos de programación (que aparecen en la columna derecha, misma fila). Tanto las nociones de mecánica como las de programación están integradas en una secuencia didáctica espiralada de complejidad creciente.

Tabla 1. Secuencia didáctica de la actividad.

Tiempo	Descripción	Conceptos de programación	Materiales
10'-15'	**Presentación de los integrantes** del equipo con sus experiencias y saberes en general.		
10'-15'	**Presentación de la actividad** **1.** Delinear la modalidad del espacio de trabajo: sugerencias para trabajar en equipo. **2.** Introducir los conceptos de robótica y programación. **3.** Reflexionar sobre posibles usos de los robots en ingeniería. **4.** Presentar la Interfaz de programación.		Consigna escrita describiendo la actividad. Programa Minibloq (programación en bloques).
135'	**Práctica (Consignas y objetivos)** *Consigna 1: Mover el robot en línea recta por un tiempo*	*Variable*	Robot Multiplo N6 de RobotGroup

	Objetivos: - Experimentar con el movimiento del robot dentro de una línea recta. - Probar distintas velocidades. - Observar el problema de la diferencia de velocidad entre ruedas. *Consigna 2: Realizar ajuste para que ambas ruedas frenen al mismo tiempo*. *Consigna 3: Programar y correr un algoritmo para avanzar hasta un obstáculo y marcar el momento de encuentro (giro 180 grados).* - Intentar realizar una medición indirecta a partir de la detección de la línea. - Reflexionar sobre qué papel juega el orden de la secuencia y las estructuras de control en el programa. *Consigna 4: Programar un algoritmo para que el robot siga una línea*.	*Proceso* *Fin de proceso* *Delay* *Condición lógica* *Bifurcación* *Iteración* *Condición de fin* *Secuencia* *Estructuras de control*	ensamblado en Fab Lab UNR (Figura 1).
15’	**Cierre de la actividad**: Reflexión final. Puesta en común de la actividad. Reflexionar acerca de otras aplicaciones posibles de este tipo de tecnología, pensar como fue el desarrollo de la actividad, que aprendieron, etc. Encuesta sobre la actividad para realizar en sus casas.		

A continuación, se describe con mayor detalle cada una de las fases de la actividad descriptas en la tabla.

Presentación de los integrantes del equipo

Cuando llegaron los participantes al laboratorio pasaron a la sala, se sentaron alrededor de la mesa de charla y se presentaron. Se les pidió que contaran brevemente los motivos para elegir ingeniería mecánica como carrera y para elegir la actividad de robótica en particular. Los profesores se presentaron y hablaron de su motivación hacia la robótica.

Presentación de la actividad

Delinear la modalidad del espacio de trabajo: sugerencias para trabajar en equipo. Luego de las presentaciones el profesor explicó cómo se desarrollaría la actividad de programación del robot en equipo. Mencionó ciertas recomendaciones para la comunicación: (a) La opinión de cada miembro del equipo es igualmente valiosa. (b) La idea que aporta una única persona puede no ser suficiente para resolver un problema complejo pero, a veces, se enriquece con el aporte de otra persona, por eso es importante que todos participen. (c) Para poder escuchar a todos es necesario respetar los turnos en la conversación.

Introducir los conceptos de robótica y programación. Para aprovechar las ventajas de aprender a programar un robot, era necesario mostrar las relaciones entre la programación y la robótica a partir de las nociones que los estudiantes ya poseían. Para ello, los profesores preguntaron al grupo qué idea tenían de *robot*, si conocían ejemplos de robots y cuáles eran características que hacen que un dispositivo sea considerado un

robot. Se intentó ir rescatando aquellos caracteres definitorios de un robot relacionándolos con nociones básicas de algorítmica (y anotándolos en el pizarrón), a saber: (a) *Movimiento*: Todo robot puede producir movimiento a través de motores o actuadores. Este movimiento, generado por programa, será un "resultado" a observar y depurar en caso de que no sea el resultado esperado. También puede interpretarse como "salida" de información. (b) *Autonomía*: el robot puede realizar ciertas tareas sin la intervención directa del usuario, precisamente por estar controlado por programa. (c) *Programa*: Un conjunto de instrucciones en la memoria del robot que podrán ser modificadas por el programador. (d) *Sensores* para captar información que *ingresa* al programa, en base a las cuales se podrá determinar qué acciones realizar. Esta información se almacena en variables. (e) *Comunicación con el entorno*: Así como los sensores "ingresan" información al programa, los motores son activados por instrucciones que "salen" del programa. El programa comunica al robot con el entorno porque contiene instrucciones de entrada y salida de información.

Presentación de la interfaz de programación. Se presentó Minibloq, una plataforma visual de programación para Arduino y otros dispositivos de robótica que permite usar bloques en lugar de líneas de código (Figura 4), no obstante, cuenta con un generador de código que permite a los estudiantes observar en tiempo real cómo se escriben las instrucciones para cada bloque.

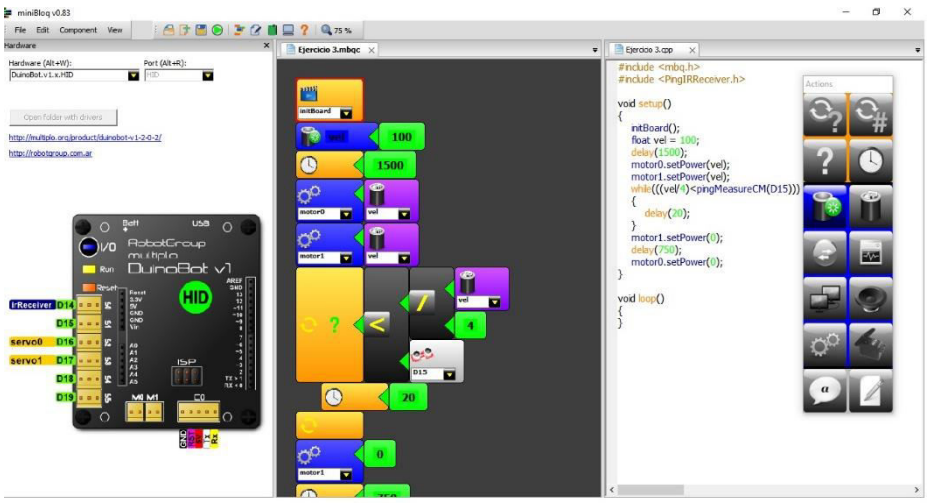

Fig. 4. Plataforma Minibloq: A la izquierda, Vista de Hardware; en el centro, editor de Minibloq; a la derecha, editor de código.

En esta etapa del encuentro, los participantes se trasladaron hacia la pantalla del ordenador ubicado cerca de la mesa de charla. Se presentó el entorno de Minibloq, las tres vistas, y las herramientas. Se recomendó ubicar el mouse sobre los distintos componentes del panel para que se active la leyenda que indica para qué se usa cada componente.

Fase de práctica

Experimentar con el movimiento del robot dentro de una línea recta. Conceptos como variable, asignación, inicialización, entre otros, se comprenden más fácilmente cuando pueden materializarse. En primer lugar se instó a los estudiantes a que identificaran qué dispositivos tenía el robot que estaba sobre la mesa, cuál era el nombre de cada uno (se pidió a los estudiantes que observaran los nombres escritos en la placa) y cuáles eran sus características. Se identificaron los tres sensores y los dos motores. Se explicó que los sensores de la parte inferior del robot FabLab eran sensores infrarrojos para medir la cantidad de luz reflejada en la superficie, y que dicha medición es un valor alojado en una variable. Por otro lado, los sensores ultrasónicos de proximidad miden la distancia al objeto frente al robot, otro valor almacenado. Se mostró cómo la placa se enchufa para cargar el programa que se realiza en la PC con Minibloq.

La primera consigna fue mover el robot en línea recta por un tiempo. Se preguntó a los estudiantes cómo creían que se movía el robot para comenzar a pensar cómo programar esa consigna. Con "línea recta" no nos referíamos a evitar las desviaciones sino simplemente activar los motores "por unos segundos". Esta mención del tiempo fue una restricción deliberada. Una idea equívoca de los estudiantes que se inician en programación (reforzada por los lenguajes visuales con bucles sin fin, o el uso de Apps en ventanas sin cierre en celulares), suele ser la de que los procesos iniciados no necesitan finalizarse. Estos motores en funcionamiento, por el contrario, sirvieron para ejemplificar la noción de proceso, que se vincula estrechamente con la idea de *fin del proceso.* Es decir, la activación de un proceso muy simple (en este caso el movimiento del motor) trajo a colación el requisito de finalización. En este primer caso, esa finalización se programó con una simple demora utilizando la instrucción *delay*, para marcar el tiempo que el programa estará demorado antes de ejecutar la instrucción siguiente, que deberá ser una instrucción de apagado de los motores.

Aprovechando la consigna, el profesor explicó el concepto de *variable* asociado a los sensores y motores: cada motor o sensor posee un espacio de memoria en el cual se "guarda" un número que puede conocerse o modificarse. Se escribió en el pizarrón la forma habitual de una asignación de variables. Por ejemplo, si *M1* es la variable del primer motor, *M1 = 50* estaría asignando el 50 por ciento de potencia al motor para que encienda a una velocidad media. Los valores de potencia asignados a un motor son porcentajes que van de 0 (apagado) a 100 (velocidad máxima posible para ese motor), es decir, *M1 = 0* significa apagar el motor (encender es asignar un número distinto de 0). En el programa simplemente se debe dar clic en el panel sobre el símbolo de motor (engranajes en la versión 0.82 Beta), y asignarle un valor de potencia.

Probar distintas velocidades. Se mostró a los estudiantes como establecer mediante programa distintas potencias en los motores y luego observaron el recorrido real del robot. Al finalizar el recorrido, en cada prueba de velocidad, se observó un desfasaje entre ruedas. Surgió la pregunta sobre qué podría estar ocasionando esa diferencia y ese movimiento de desviación al final. Los estudiantes propusieron posibles causas, en algún punto, dependiendo del grupo, surgía la inquietud "¿Qué efecto tendría en el robot si una rueda rotase más rápidamente que otra? El efecto observado era que,

efectivamente, el robot daría un giro. Dado que la consigna 1 indicaba que el robot debía avanzar *en línea recta* por un tiempo, había que evitar dicho giro, logrando que los ejes de ambas ruedas fueran a la par.

Consigna 2: *Realizar ajuste para que ambas ruedas frenen al mismo tiempo*. Se aprovechó esta instancia para introducir nociones de *predicción y control*. En primer lugar, se preguntó cómo es posible averiguar qué es lo que ocasiona ese giro. Se puede comenzar descartando de a una las *posibles causas*, por ejemplo, averiguar si se trata de algún desperfecto en el piso, trasladar el robot a otro tipo de piso dentro del establecimiento (variar las condiciones) y observar nuevamente la trayectoria. Si el error persiste, se descarta que la causa sea el piso. Luego de descartar otras posibilidades, se llegó a la conclusión de que podría tratarse de un inconveniente mecánico que hace que uno de los motores funcione por más tiempo. Como en otras instancias, para estimular la participación se esperó que los estudiantes realicen sugerencias antes de que las planteen los profesores. Los estudiantes propusieron varias formas de medir la diferencia entre las ruedas, filmarla para medir el tiempo en el video, medir la diferencia en el espacio para averiguar el tiempo de ese recorrido, entre otras. Se preguntó a los estudiantes cómo corregir el programa para que el motor que se detenía antes, funcionara unas décimas de segundos más. La solución más común fue, "encender los motores, demorar 3 segundos, conociendo la diferencia entre ruedas, detener primero el motor de recorrido más extenso, e introducir una demora (igual a la diferencia) antes de apagar el motor de recorrido menor" (Figura 5).

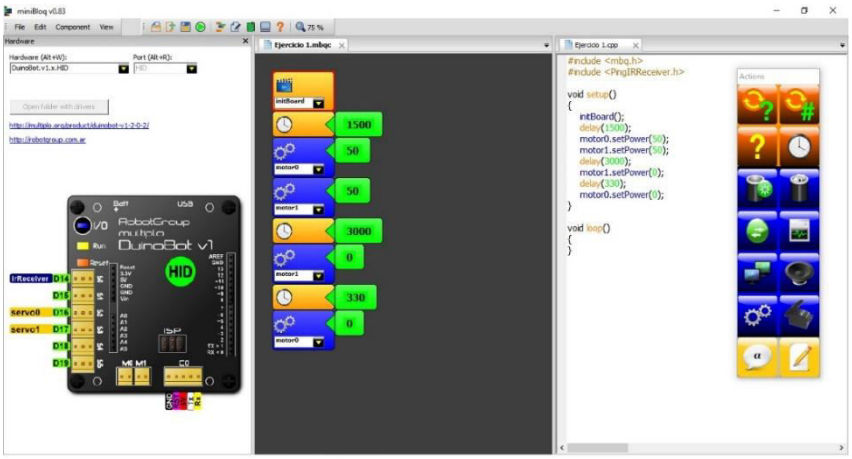

Fig. 5. Código en Minibloq para encendido de motores y retraso de una rueda.

Consigna 3: Programar y correr un algoritmo para avanzar hasta un obstáculo y marcar el momento de encuentro (giro 90º). En esta consigna se utiliza el sensor ultrasónico para detectar la cercanía de objetos. Se vuelve a mencionar el concepto de variable ya que el sensor de proximidad tenía asociada la variable S1, donde se almacena un número que indica la distancia hasta el objeto más próximo calculada por ultrasonido. Aquí surgió la segunda posibilidad de establecer una condición de fin (la primera mención, más simple, fue en la consigna 1), en este caso más sofisticada,

dependiente de la información almacenada en la variable del sensor S1. Se estableció la *condición* de que si se detecta que la distancia próxima es menor que un valor mínimo (por ejemplo, si se acerca a una pared, o si alguien coloca una mano enfrente del robot), el robot debería detenerse para no chocar. Se explicó la estructura de esta bifurcación condicional, la limitación de preguntar sólo una vez por la condición, y el reemplazo de la bifurcación simple por una iteración que reitere la pregunta por la cercanía del obstáculo hasta que éste aparezca.

El robot debe detenerse cuando encuentre un objeto en el frente a una distancia mínima a establecer por programa. Se introducen los *conceptos de iteración* (bucle) y se vuelve a mencionar la noción de condición lógica pero, esta vez, asociada a una iteración para que el sensor repita constantemente la lectura de la distancia hasta que esa distancia sea menor o igual a la mínima establecida.

Consigna 4: Programar un algoritmo para que el robot siga una línea. En esta consigna se complejiza más el uso de las condiciones lógicas. Si el robot comienza el recorrido desde una línea, se debe chequear si ambos sensores están sobre la línea, o caso contrario, averiguar cuál de los dos sensores se movió hacia afuera para volver a posicionarlo en la línea.

En este punto, se pudo reflexionar sobre qué papel juega el orden de la secuencia y las estructuras de control en el programa, así como observar la complejidad de las estructuras de condición anidadas.

Se explicó a los estudiantes que el sensor infrarrojo que está debajo del robot, funciona emitiendo luz con un diodo emisor de infrarrojo y recibiendo luz por rebote con un fototransistor. Si lo que está "observando" es una superficie oscura, la superficie absorbe mayor cantidad de luz y rebota menos luz hacia el fototransistor. Cuando la *variable* del sensor infrarrojo indicara un valor bajo, podría estar frente a una línea. Se colocó una cinta aisladora negra para dibujar la línea en el piso.

Dado que el programa anterior tenía una lógica similar que incluía una iteración para la detección de obstáculos, se reutilizó esa estructura cambiando el sensor y la condición de fin para la detección de una línea en el piso.

3.2 Recolección de datos

Al finalizar cada uno de los encuentros, cada integrante del equipo escribió sus observaciones que fueron compartidas semanalmente en reuniones presenciales, en intercambios informales a distancia (telefónicos o vía mail) y en una plataforma en google drive donde paulatinamente se subieron los registros que pudieron ser leídos por todos los docentes durante el proceso. A partir de las observaciones se ajustaron condiciones de los encuentros. Al finalizar el taller, se recolectaron las últimas encuestas y se compartió el archivo con las respuestas para realizar el análisis.

3.3 Primeros resultados en torno a las percepciones generales

Al finalizar la actividad, los participantes completaron desde su domicilio una encuesta acerca de sus impresiones sobre el encuentro basada en las variables que se mencionaron en la sección *Método* y se explicitan a continuación (Tabla 2).

Tabla 2. Preguntas sobre la actividad de robótica.

Orden	Pregunta
1	¿Cuál experiencia *maker* realizó?
2	¿La consigna propuesta fue clara?
3	¿La metodología de trabajo fue clara?
4	¿El espacio físico piensa que fue adecuado?
5	¿Qué opinión tiene en torno a los elementos utilizados (herramientas, materias primas, instrumentos, etc.)?
6	¿Cuáles conceptos de otras asignaturas pudieron integrar en esta experiencia?
7	¿Qué habilidades creativas pudo poner en juego en esta experiencia?
8	¿Qué criterios funcionales propuso en la resolución del problema?
9	¿Cómo fue la relación con sus compañeros en el marco de la experiencia?
10	¿Qué nos sugieren para una óptima conformación del grupo?
11	¿Cómo fue la relación con los docentes en el marco de la experiencia?

Las preguntas 2 a 5 hacen referencia a la percepción general del encuentro en relación a la consigna, metodología, espacio y herramientas. Las preguntas 9 y 11 también indagan la percepción general pero vinculada a las relaciones personales con compañeros y docentes. El grado de acuerdo del alumno en todas las preguntas sobre percepción se expresó con una escala de 1 a 5. Por otro lado, las preguntas 6 a 8 son metacognitivas, implican una reflexión del estudiante acerca de su propio proceso de aprendizaje en relación a la carrera, la pregunta 6 en particular refiere a los conocimientos previos. Estos tres ítems eran preguntas abiertas para habilitar una mayor especificidad en la respuesta. En la Figura 6 se exponen los resultados de las preguntas 2 a 5.

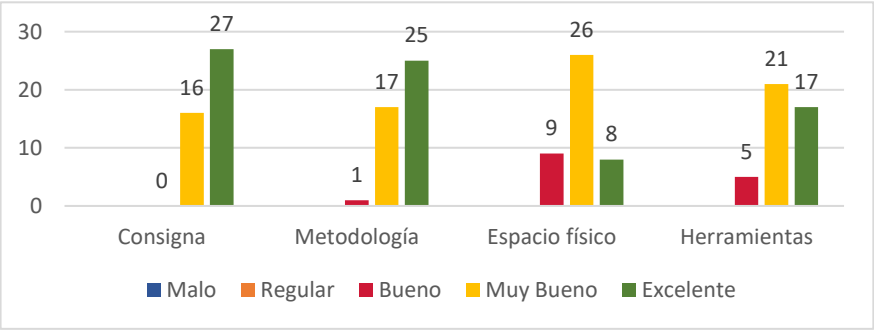

Fig. 6. Resultados sobre percepción de Consigna, Metodología de trabajo, Espacio físico y Herramientas utilizadas

Sobre el total de respuestas en torno a la percepción se observa que ningún estudiante calificó con la categoría mala o regular. Los resultados mejor calificados estuvieron en torno a la Consigna y Metodología de trabajo, mientras que el Espacio físico y Herramientas utilizadas obtuvieron una puntuación menor. Entendemos que esto se debe por una parte, a que el espacio físico utilizado, todavía requiere ajustes por una mudanza retrasada, y en torno a las herramientas el equipo manipulado es antiguo y existen robots más modernos, que los mismos estudiantes señalaron.

En la Figura 7 se exponen los resultados de las preguntas 9 y 11.

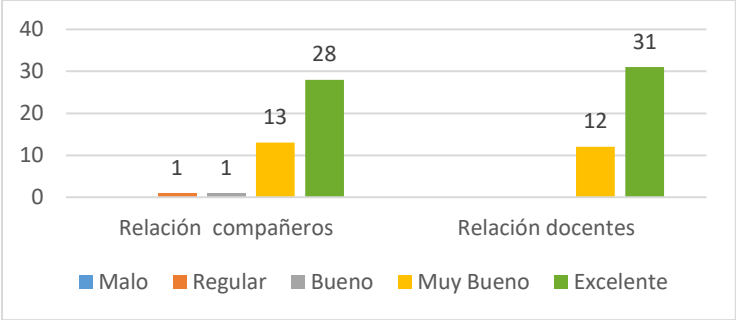

Fig. 7. Percepción de la relación con compañeros y con docentes.

Las respuestas reflejan el buen clima de trabajo entre compañeros y con los docentes, la buena disposición a colaborar y el respeto mutuo durante la actividad.

En la *pregunta 6* de la encuesta se solicitó a los alumnos que mencionaran los "conocimientos previos que habían podido integrar" en la actividad. Las experiencias estaban planteadas de modo tal que los estudiantes pudieran integrar saberes más básicos. Sólo 1 alumno de 51 (2%) manifestó no haber podido integrar conocimientos de otras asignaturas. El 98% restante mencionó varios conceptos sumando un total de 73 menciones (N).

Se revisaron las 50 respuestas en busca de categorizar los conocimientos previos mencionados. Como se utilizaron términos distintos para un mismo tema, éstos se agruparon en categorías, por ejemplo, algunos estudiantes mencionaron que pudieron integrar conceptos de "Informática", otros de "lógica de programación", algunos de "programación", todos estos casos se englobaron en la categoría "Informática, lógica y programación". En la categoría Física se incluyeron todas las referencias a "velocidad", "fuerza", "movimiento" entre otras. Las categorías "Cálculo", "Álgebra", "Geometría", "Química" y "Electricidad" no engloban más que el propio término.

En la Tabla 3, puede verse que se generaron en total 73 menciones a conceptos previos, y se clasificaron dentro de 6 categorías, todos ellos correspondientes a asignaturas o contenidos fundamentales del ciclo de ingreso, el mayor porcentaje de los cuales corresponde a Informática, lógica y programación (43,84%), seguido por Física

(20,55%), Geometría (6,85%), Álgebra (12,33%), Cálculo (15,07%), Electricidad (1,37%).

Tabla 3. Integración de conceptos por actividad

N=73	Informática, lógica y programación	Física	Geometría	Álgebra	Cálculo	Electricidad
Frecuencia absoluta	32	15	5	9	11	1
Frecuencia porcentual	43,84	20,55	6,85	12,33	15,07	1,37

Con respecto a la pregunta 7 "¿Qué habilidades creativas pudo poner en juego en esta experiencia?", la mayoría de los comentarios se enfocaron en la creatividad para la resolución de problemas, en menor medida se mencionó la posibilidad de ser creativos "para trabajar en equipo" y para organizar la actividad.

La pregunta 8 "¿Qué criterios funcionales propuso en la resolución del problema?", fue interpretada en su mayoría como qué funciones debía desempeñar el robot, y se mencionaron varias: funciones lógicas dentro del mismo programa, acciones como "doblar a la izquierda", "evitar obstáculos". Pero también hubo mención de funciones más generales de la consigna como "prueba y error", "interpretación", "ordenamiento correcto de los pasos", "razonamiento" y "trabajo en equipo".

3.4 Registro de observaciones y bitácoras

En general, las observaciones están en línea con las encuestas. Los docentes han notado una gran implicación de los alumnos en la realización de la tarea de programar el robot y verlo en funcionamiento: "No consultaron nunca el celular. Algo que suelen hacer con frecuencia durante las clases tradicionales".

De varias maneras los estudiantes han manifestado que hay algo especialmente novedoso y motivador en el hecho de poder "ver" físicamente lo que programaron. Algunos de estos estudiantes ya han programado en otros lenguajes. Otro punto que se reitera por parte de los estudiantes es la solicitud de "más tiempo" y de que estos encuentros "se repitan", algo que no estuvo al alcance de la cátedra hasta el momento.

4 Discusión y conclusiones

En marzo de 2022, ACM, la organización sobre ciencias de la computación de mayor envergadura a nivel internacional, en el journal sobre educación (TOCE), dedicó un número especial a resaltar la importancia de la conceptualización y el uso de teoría *educativa* en las investigaciones *educativas* en ciencias de la computación [58]. Esta preocupación ya había sido manifestada por Fincher y Petre [59], y representada

mediante un cuadrante de cuatro extremos y dos dimensiones (teoría y experiencia): en la parte superior izquierda ubicaron las investigaciones con mucha argumentación teórica pero ninguna relación con la experiencia, abajo a la derecha, los documentos que se basan principalmente en la experiencia, pero con poca o ninguna argumentación teórica, señalando que precisamente éste es el tipo de trabajo más frecuente en educación informática y que el tipo de investigaciones menos frecuentes, son las que ubicaron en la parte superior derecha, las que tienen buena argumentación teórica y buena relación con la experiencia. Este trabajo constituye un esfuerzo por alcanzar esa meta. Brindar un marco sólido en teorías de la educación que pueda aplicarse a una experiencia concreta de aprendizaje de programación. Implicó establecer un diálogo respetuoso entre expertos del área de ingeniería y de las ciencias de la educación para que las teorías se materialicen en soluciones prácticas a problemas reales del estudiantado.

En relación al estado actual de la educación en ingeniería, hasta el momento, se percibe una preponderancia de los enfoques situados -tales como el aprendizaje basado en problemas- y poca presencia de aproximaciones *cognitivas*. Algunas nociones fundamentales como el rol de la abstracción en el aprendizaje, y la diferencia entre conocimiento *declarativo* y *procedimental*, solo tienen cabida en los enfoques cognitivos, por lo cual se sugiere incorporarlos como complementarios de otras perspectivas.

Con respecto a la *abstracción* como requisito fundamental en informática, es ampliamente aceptada la dificultad de los estudiantes para manipular tanto abstracciones de datos como de proceso [60]–[63], y su tendencia a reducir el nivel de abstracción hacia componentes concretos [64]–[67]. Al ofrecer elementos concretos que exteriorizan comportamiento se materializa y facilita la explicación de operaciones abstractas, tal como han propuesto los diseñadores de lenguajes visuales [68], [69]. Una ventaja de utilizar robots para enseñar a programar, tal como lo han manifestado claramente los estudiantes, es la posibilidad de hacer "visible" la ejecución del algoritmo en cada paso del movimiento del robot y de facilitar al estudiante la representación de estructuras lógicas. Cada línea de código que escriben en el programa es el resultado de imaginar la "conducta" del robot anticipando mentalmente posibles movimientos que habrán de constatarse visualmente.

Los ejemplos típicamente utilizados en las carreras de informática para aprender a programar suelen provenir de la matemática, la física, la administración, u otras disciplinas, cuyos resultados son verificables en un único dispositivo de salida: la pantalla. Es esperable que el resultado final de estos algoritmos sea un número (o varios) y que para corroborar su exactitud se deba conocer la operatoria del problema en su totalidad, con lo cual, no sólo el estudiante debe aprender las estructuras de control de programación sino también las abstracciones propias del problema. Es difícil distinguir entre los errores por desconocimiento de la programación o los que ocurren por desconocimiento del problema. El problema del movimiento de un objeto (el robot), observable a simple vista, no exige un nuevo saber disciplinar al estudiante, salvo el conocimiento de la existencia de variables en los sensores y otros mecanismos simples que fueron proporcionados como parte del andamiaje. Tómese el caso de la *iteración*, una estructura de control que suele ser ejemplificada con secuencias de datos, como

"Ingresar una secuencia números naturales y calcular su cuadrado. Considerar el cero como fin de datos". Para comprender el proceso de iteración, incluyendo la condición de fin, el estudiante debe ser capaz de imaginar la secuencia de números naturales ingresando por teclado, a los cuales se aplica un tratamiento, sin olvidar que uno de esos números podría ser un cero y marcaría el fin de la secuencia. Los elementos de la secuencia (números) son abstracciones. En cambio, la repetición del giro de la rueda de un motor hasta que encuentre un obstáculo (condición de fin), es más intuitiva por basarse en elementos perceptuales que facilitan su representación mental. Por otra parte, la motivación por depurar y verificar aumenta cuando hay poca distancia temporal entre codificación y prueba (como sucede con los lenguajes interpretados). Dicho esto, se infiere que la robótica no sólo es idónea para introducir a estudiantes de ingeniería mecánica a la programación, sino también a estudiantes de otras ingenierías.

Dado que mediará un tiempo entre el taller de robótica que se implementó en el *Fab Lab* y la asignatura Informática en la carrera de ingeniería mecánica, una debilidad de esta investigación podría considerarse la escasa disponibilidad de recursos para extender el taller a 3 encuentros como mínimo, en los cuales pudieran reforzarse los mismos conceptos con diferentes ejemplos, con una evaluación más minuciosa y explorar otros puntos pendientes, como por ejemplo, qué papel juega el ocultamiento de la sintaxis en los lenguajes visuales, ya que puede facilitar la comprensión de código complejo o complejizar la comprensión de primitivas simples. Lenguajes como Minibloq permiten visualizar ambas opciones y los estudiantes con experiencia en programación, no siempre eligen la implementación visual.

A partir de las respuestas en las encuestas, las bitácoras docentes y la observación de los investigadores, pudo constatarse que la mayoría de los alumnos logró integrar conceptos de otras asignaturas. Esto es un logro importante, teniendo en cuenta las limitaciones de tiempo y recursos. Se espera que con una mayor disponibilidad de ambos, en el futuro, podrían enriquecerse y reforzarse tanto los aprendizajes como la motivación.

La evidencia de conceptos previos integrados a partir de una didáctica espiralada que alterna momentos de conceptualización y de puesta a prueba, confirma que la mediación del docente en el trabajo de taller es fundamental para un aprendizaje significativo en *Fab Lab*. Tal como se expresó anteriormente "el profesor acompaña a construir el *concepto*, no sólo el objeto".

Los resultados muestran que la percepción general de la experiencia de robótica fue positiva. Esto se evidenció también durante los encuentros en la participación activa de los estudiantes y su voluntad de colaborar en todas las consignas. Los programas en Minibloq generados y archivados mostraron un alto nivel de variabilidad entre sí. Lo que permite vislumbrar que las soluciones fueron creativas, no copias ni automatismos aprendidos. Esto también puede asociarse al trabajo en equipo, ya que cada solución individual debía ser evaluada y eventualmente modificada por el grupo, y luego revisada en las sucesivas instancias de programación.

Advertimos que en relación a la adquisición de conceptos habría sido productivo compartir en el campus un resumen de la experiencia explicitando los conceptos aprendidos y los programas generados por cada grupo, luego del cierre de la actividad, aunque estimamos conveniente no dar a conocer públicamente todas las actividades

para mantener el efecto de novedad que sostiene la curiosidad (y la motivación) durante el encuentro.

Se deduce que la secuencia didáctica resultó apropiada para trabajar con ingresantes, lo que confirma el potencial de estas prácticas de taller para aprender programación en el ciclo introductorio de ingeniería mecánica y extender la propuesta a otras ingenierías.

5 Referencias

1. CONFEDI, «Propuesta de estándares de segunda generación para la acreditación de carreras de ingeniería en la República Argentina», *Aprob. Por Asam. Cons. Fed. Decanos Ing. Repúb. Argent. Rosario*, vol. 1, 2018.
2. K. C. Ferrando, «Perfil del Ingeniero y formación complementaria en carreras de Ingeniería», 2014.
3. P. V. Paoloni, A. C. Chiecher, y R. C. Elisondo, «Graduados de ingeniería y competencias genéricas. Cinco estudios de la última década que recuperan sus valoraciones y experiencias», *Rev. Educ. En Ing.*, vol. 14, n.º 28, pp. 54-64, 2019.
4. G. Rodríguez, M. Raposo, F. Sklate, y P. Demartini, «Análisis de experiencias de una cátedra de Introducción a la Ingeniería Mecánica», p. 11, 2018.
5. «Fab Lab UNR», *Fab Labs.io - The Fab Lab Network*, 2022. https://www.Fab Labs.io/labs/Fab Labunr (accedido 13 de junio de 2022).
6. P. S. San Martín, «Dispositivos Hipermediales Dinámicos», p. 5, 2011.
7. Andrés, G. D., & San Martín, P. S. (2019). Modelo analítico multidimensional para la construcción y la evaluación de prácticas educativas mediatizadas en Educación Superior. RAES: Revista Argentina de Educación Superior, 18, 88-104.
8. Baquero, R. (2017). Desarrollo subjetivo, prácticas educativas y prácticas escolares. Los enfoques socioculturales como herramienta de análisis. Revista de Didáctica Psicología Pedagógica Uberlândia, 1(2), 291-309.
9. Albanese, M. A., & Mitchell, S. (1993). Problem-based learning: A review of literature on its outcomes and implementation issues. Academic Medicine-Philadelphia-, 68, 52-52.
10. Barrows, H. S. (1996). Problem-based learning in medicine and beyond: A brief overview. New directions for teaching and learning, 1996(68), 3-12.
11. Bruner, J. (1966). Toward a theory of instruction (Vol. 59). Harvard University Press.
12. Bruner, J. (1960). The process of education (pp. xvi, 97). Harvard Univer. Press.
13. Ausubel, D. P., Novak, J. D., & Hanesian, H. (1968). Educational psychology: A cognitive view (Vol. 6). holt, rinehart and Winston New York.
14. Reigeluth, C. M., & Rodgers, C. A. (1980). The elaboration theory of instruction: Prescriptions for task analysis and design. NSPI journal, 19(1), 16-26.
15. Rodríguez Moneo, M. El proceso de enseñanza y aprendizaje de competencias. En *Evaluación global de los resultados del aprendizaje en las titulaciones dentro del Espacio Europeo de Educación Superior* 19–44 (Dykinson, 2011).
16. Aparicio, J. & Rodríguez-Moneo, M. *El aprendizaje humano y la memoria. Una visión integrada y su correlato neurofisiológico*. (Pirámide, 2015).
17. Pozo, J. I., Pérez, M., Dominguez, J., Gómez, M. Á., & Postigo, Y. (1994). La solución de problemas. Santillana.

18. Çakıroğlu, Ü., & Öztürk, M. (2018). Evaluating an Online Programming Instructional Process Organized Through Elaboration Theory. International Journal of Web-Based Learning and Teaching Technologies (IJWLTT), 13(4), 1-16. https://doi.org/10.4018/IJWLTT.2018100101
19. Bull, G., Garofalo, J., Littman, M., Sherman, R., Hoffman, M., Grant, M. M., & Grier, A. (2017). Make to learn: Invention through emulation. Smart Learning Environments, 4(1), 8. https://doi.org/10.1186/s40561-017-0047-5
20. Bosse, Y., & Gerosa, M. A. (2017). Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. ACM SIGSOFT Software Engineering Notes, 41(6), 1-6. https://doi.org/10.1145/3011286.3011301
21. Jenkins, T. (2002). On the Difficulty of Learning to Program. 1-8.
22. D. MacKenzie y J. Wajcman, *The social shaping of technology*. Buckingham, UK: Open University Press, 1999. Accedido: 28 de abril de 2022. [En línea]. Disponible en: http://mcgraw-hill.co.uk/openup/
23. P. S. San Martín, «Aspectos sociales y tecnológicos del Dispositivo Hipermedial Dinámico desarrollados en diferentes contextos educativos», *Rev. Educ.*, vol. 5, n.º 5, pp. 81-98, 2013.
24. J. van Dijk, *The Digital Divide*. John Wiley & Sons, 2020.
25. J. A. Van Dijk, «Digital divide research, achievements and shortcomings», *Poetics*, vol. 34, n.º 4-5, pp. 221-235, 2006.
26. J. A. Van Dijk, «Digital divide: Impact of access», *Int. Encycl. Media Eff.*, pp. 1-11, 2017.
27. Vygotsky, L. S. & Cole, M. *Mind in society: Development of higher psychological processes*. (Harvard University Press, 1978).
28. Rivière, Á. Desarrollo y educación: El papel de la educación en el "diseño" del desarrollo humano [Development and education. The role of education in the 'desing' of human development]. in *Ángel Rivière. Obras escogidas* (eds. Belinchón, M., Rosa, A., Sotillo, M. & Marichalar, I.) vol. 3 203–242 (Panamericana, 2003).
29. K. Koedinger, E. McLaughlin, J. Kim, J. Jia, y N. Bier, «Learning is Not a Spectator Sport: Doing is Better than Watching for Learning from a MOOC», pp. 111-120, mar. 2015, doi: 10.1145/2724660.2724681.
30. G. E. Baykal, M. Van Mechelen, M.-L. Wagner, y E. Eriksson, «What *FabLearn* talks about when talking about reflection—A systematic literature review», *Int. J. Child-Comput. Interact.*, vol. 28, p. 100256, 2021.
31. P. Blikstein y D. Krannich, «The makers' movement and Fab Labs in education: experiences, technologies, and research», en *Proceedings of the 12th international conference on interaction design and children*, 2013, pp. 613-616.
32. D. Scaradozzi, L. Guasti, M. Di Stasio, B. Miotti, A. Monteriù, y P. Blikstein, *Makers at School, Educational Robotics and Innovative Learning Environments: Research and Experiences from FabLearn Italy 2019, in the Italian Schools and Beyond*. Springer Nature, 2021.
33. P. Blikstein, «Maker movement in education: History and prospects», *Handb. Technol. Educ.*, vol. 419, p. 437, 2018.

34. C. Fernandez, T. Hochgreb-Haegele, y P. Blikstein, «From “Playful Activities” to “Knowledge Building”: A Case Study about a Teacher’s Percpetions on the Role of Experiments», 2021.
35. J. A. Valente y P. Blikstein, «Maker education: Where is the knowledge construction?», *Constr. Found.*, vol. 14, n.º 3, pp. 252-262, 2019.
36. E. Von Glasersfeld, *Radical constructivism: a way of knowing and learning*. London ; Washington, D.C: Falmer Press, 1995.
37. E. Von Glasersfeld, «An introduction to radical constructivism», *Inven. Real.*, vol. 1740, p. 28, 1984.
38. S. Papert y I. Harel, «Situating constructionism», *Constructionism*, vol. 36, n.º 2, pp. 1-11, 1991.
39. P. Freire, *Pedagogía del oprimido*. Siglo xxi, 2005.
40. I. Illich, «La sociedad desescolarizada». Ediciones Godot Buenos Aires, Argentina, 2011.
41. D. Kokotsaki, V. Menzies, y A. Wiggins, «Project-based learning: A review of the literature», *Improv. Sch.*, vol. 19, n.º 3, pp. 267-277, nov. 2016, doi: 10.1177/1365480216659733.
42. S. Papert, «What is Logo? Who needs it», en *Logo philosophy and implementation*, Logo Computer Syst., Inc., 1999, pp. 4-16.
43. M. Resnick *et al.*, «Scratch: programming for all», *Commun. ACM*, vol. 52, n.º 11, pp. 60-67, 2009.
44. J. Maloney, M. Resnick, N. Rusk, B. Silverman, y E. Eastmond, «The scratch programming language and environment», *ACM Trans. Comput. Educ. TOCE*, vol. 10, n.º 4, pp. 1-15, 2010.
45. U. Wilensky, «NetLogo. Evanston, IL: Center for connected learning and computer-based modeling, Northwestern University». 1999.
46. N. G. Monjelat, G. L. Rodriguez, y P. S. S. S. Martín, «Modelado y simulación de un sistema mecánico simple: Programar en primer año de ingeniería», *Rev. Educ. En Ing.*, vol. 13, n.º 25, Art. n.º 25, feb. 2018, doi: 10.26507/rei.v13n25.796.
47. Reigeluth, C. M. (1983). Instructional-design Theories and Models: An overview of their current status. Psychology Press.
48. Reigeluth, C. M. (2016). Instructional theory and technology for the new paradigm of education. Revista de Educación a Distancia (RED), 50.
49. J. Wing, «Computational thinking», *J. Comput. Sci. Coll.*, vol. 24, n.º 6, pp. 6-7, jun. 2009.
50. J. M. Wing, «Computational thinking», *Commun. ACM*, vol. 49, n.º 3, pp. 33-35, mar. 2006, doi: 10.1145/1118178.1118215.
51. J. R. Anderson, «Acquisition of cognitive skill.», *Psychol. Rev.*, vol. 89, n.º 4, p. 369, 1982.
52. K. A. Rawson, R. C. Thomas, y L. L. Jacoby, «The Power of Examples: Illustrative Examples Enhance Conceptual Learning of Declarative Concepts», *Educ. Psychol. Rev.*, vol. 27, n.º 3, pp. 483-504, sep. 2015, doi: 10.1007/s10648-014-9273-3.
53. A. Jamrozik y D. Gentner, «Relational labeling unlocks inert knowledge», *Cognition*, vol. 196, p. 104146, mar. 2020, doi: 10.1016/j.cognition.2019.104146.
54. V. D’Angelo y M. Trench, «Defending Diversity: Providing Examples from Different Domains Enhances Application of System Principles Beyond the

Domains Covered by the Examples.», presentado en Cognitive Science Society «Cogsci 2022: Diversity & Inclusion» (in press), Toronto, Ontario, jul. 2022.
55. E. L. Deci y R. M. Ryan, «Cognitive evaluation theory», en *Intrinsic motivation and self-determination in human behavior*, Springer, 1985, pp. 43-85.
56. «robotgroup.com.ar». http://www.robotgroup.com.ar/web/ (accedido 15 de junio de 2022).
57. «miniBloq». v/ (accedido 15 de junio de 2022).
58. J. Tenenberg y L. Malmi, «Conceptualizing and Using Theory in Computing Education Research», ACM Transactions on Computing Education, 2022.
59. S. Fincher y M. Petre, Computer Science Education Research. CRC Press, 2004.
60. M. Armoni, «On teaching abstraction in CS to novices», Journal of Computers in Mathematics and Science Teaching, vol. 32, n.o 3, pp. 265-284, 2013.
61. P. J. Denning, «Systems abstractions», Commun. ACM, vol. 65, n.o 4, pp. 22-24, abr. 2022, doi: 10.1145/3517218.
62. J. Kramer, «Is abstraction the key to computing?», Communications of the ACM, vol. 50, n.o 4, pp. 36-42, 2007.
63. O. Hazzan y J. Kramer, «The role of abstraction in software engineering», en Companion of the 30th international conference on Software engineering, 2008, pp. 1045-1046.
64. E. M. Jiménez Rey, «Computación en Ingeniería: la experiencia de pensar para solucionar problemas con algoritmos y programas en aula real y aula virtua», presentado en I Simposio Argentino de Educación en Informática (SAEI 2019) - JAIIO 48 (Salta), 2019. Accedido: 16 de diciembre de 2022. [En línea]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/88784
65. O. Hazzan, «Reducing abstraction level when learning abstract algebra concepts», Educational Studies in Mathematics, vol. 40, n.o 1, pp. 71-90, 1999.
66. O. Hazzan, «How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science», Computer Science Education, vol. 13, n.o 2, pp. 95-122, 2003.
67. O. Hazzan y R. Zazkis, «Reducing abstraction: The case of school mathematics», Educational Studies in mathematics, vol. 58, n.o 1, pp. 101-119, 2005.
68. J. Maloney, M. Resnick, N. Rusk, B. Silverman, y E. Eastmond, «The scratch programming language and environment», ACM Transactions on Computing Education (TOCE), vol. 10, n.o 4, pp. 1-15, 2010.
69. M. Resnick et al., «Scratch: programming for all», Communications of the ACM, vol. 52, n.o 11, pp. 60-67, 2009.