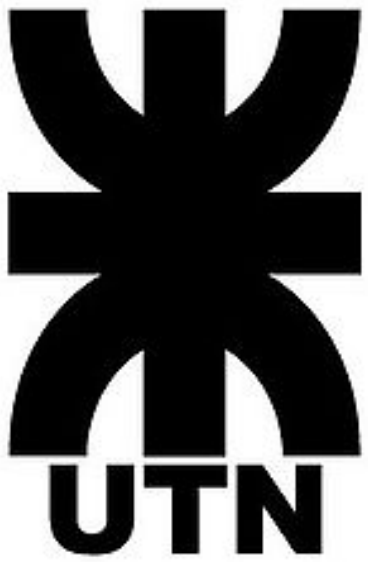


Universidad Tecnologica Nacional

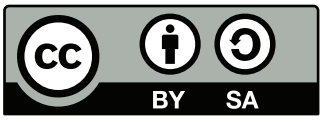
Escuela de Posgrado

Trabajo final de la carrera de licenciatura en tecnología educativa.

Proyecto de transferencia de tecnología educativa.

Implementar robótica diseñada con software y hardware libre como recurso educativo para la enseñanza de lenguajes formales.

Valentin Basel.



Programa, o seras programado.

Douglas Rushkoff.

Índice general

0.1. Introducción

Dada la creciente importancia que la tecnología digital tiene hoy en día y gracias a su transversalidad en la vida cotidiana, las TIC se han vuelto una parte integral del proceso educativo. Por consiguiente, la implementación y el desarrollo de propuestas educativas que permitan la evolución cognitiva de los discentes y el aprovechamiento de las nuevas tecnologías se vuelven de especial interés por sus ventajas como recurso educativo. En consecuencia, y como sostiene Chavarría (2011), la discusión sobre el uso de software y hardware libre como herramienta educativa, cobra especial relevancia para el desarrollo de conceptos como la soberanía tecnológica dentro del aula.

La robótica utilizada en el contexto educativo tiene múltiples aplicaciones como herramienta transversal en el aprendizaje de conceptos de matemática, física y lenguajes de programación. Tomando al construccionismo de Papert (1993) como teoría pedagógica central, podemos decir que la robótica es una herramienta construccionista (Pittí y cols., 2010), que permite a los discentes; abordar la construcción de su propio conocimiento a partir del diseño y fabricación de un robot de carácter pedagógico.

Su carácter multidisciplinario permite abordar distintas etapas de la construcción del conocimiento por parte de los discentes. Ademas, la robotica puede ser, aplicada en una gran variedad de temáticas; su espectacularidad (entendida como la capacidad de generar algún tipo de asombro en la población estudiantil), ayuda a los docentes en la tarea de impartir la curricula planteada en el curso. Sin embargo, la dificultad técnica y la cantidad de conocimientos específicos necesarios para desarrollar un robot - electrónica, mecánica y ciencias computacionales-, además del costo generalmente elevado de los kits que se consiguen en el mercado, hace que la enseñanza de robótica en las escuelas quede relegada, generalmente, a colegios que pueden financiar los costes de capacitación y adquisición de esos kits

comerciales.

A partir de lo señalado, la presente tesis propone un proyecto de transferencia educativo-tecnológico, basado en el diseño e implementación de hardware electrónico para el desarrollo de elementos de robótica, domótica o automatización, y software de control para dicho hardware, basado en los principios de soberanía tecnológica y las licencias de software libre (versión GPLV3). El objetivo general es transferir el conocimiento técnico para ”independizar” a los colegios de proveedores y fabricantes de hardware especializado en robótica educativa, preparando a los docentes y discentes en la fabricación y uso/implementación del hardware ICARO NP07 en los respectivos espacios curriculares. Dicho hardware es una herramienta pedagógica diseñada para la enseñanza de lenguajes de programación.

El proyecto ICARO busca desarrollar una solución técnica basada en hardware de especificaciones abiertas y software libre, para facilitar la labor del docente a la hora de trabajar los contenidos técnicos complejos que implica el abordaje de una disciplina como la robótica. Además, permite abaratar costos, al ser un proyecto pensado para implementarse en pequeña escala, sin equipamiento industrial y con la idea de que los docentes y discentes desarrollen (entendiendo la producción como el hecho de soldar los componentes electrónicos) el hardware ICARO. El énfasis esta puesto aqui en, (el acto de) la fabricación como una herramienta construccionista y de aprehensión, entendiendo esta práctica como una ”herramienta de liberación” en el sentido planteado por Paulo Freire (2015).

El proyecto de transferencia tecnológica educativa Bareño (2011) estará dividido en seis etapas, a lo largo de las cuales se investigará para determinar la viabilidad técnica - presupuestaria de la implementación de hardware libre en la enseñanza escolar. El proyecto contempla la capacitación de docentes y discentes para la fabricación, uso y mantenimiento del hardware propuesto. A partir del proceso de ”auto fabricación” de las placas ICARO NP07, docentes y discentes

de la institución se apropiaran de los conceptos técnicos necesarios para poder mantener y reparar el hardware y utilizarlo como una herramienta educativa, principalmente por sus posibilidades la enseñanza de lenguajes de programación. Al respecto, vale recordar que la ”programación” es considerada como una ”capacidad necesaria” segun la ley Nacional de Educación N°26.206.

Capítulo 1

Marco Teórico

1.1. Introducción

Abordar un proyecto de robótica como Recurso Educativo Abierto (REA/OER), implica una serie de desarrollos conceptuales y teóricos que van a mas allá del diseño y desarrollo técnico de hardware y software. El uso de REA/OER (Montoya y Aguilar, 2012) implica un compromiso político-epistemológico con respecto al uso de la tecnología, esto es; la apropiación de términos conceptuales como ”software libre”, ”soberanía tecnológica” y cultura libre. En lo que sigue, se da cuenta de los conceptos técnicos/teóricos para la implementación de un proyecto de robótica educativa basado en software y hardware libre.

1.2. Construccionismo

El construccionismo es una teoría del aprendizaje diseñada por Seymour Papert (1987) donde se resalta la importancia de la acción como parte principal en el proceso de aprendizaje. Toma como inspiración las ideas de la psicología constructivista, en especial el supuesto de que el conocimiento debe ser construido por

el propio sujeto, el cual aprende a través de la acción. Por lo tanto, el conocimineto no es algo que se pueda solamente ”transmitir”.

La teoría del construccionismo afirma que el aprendizaje es mucho mejor cuando los sujetos se comprometen en la construcción de un producto significativo para ellos, como podría ser un programa de computación, un robot, un juguete o una canción.

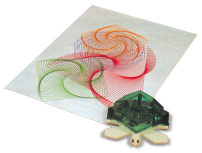


Figura 1.1: robot ”tortuga” de la empresa Valiant

De esta forma, esta teoria involucra dos tipos de construcción: cuando los sujetos construyen cosas en el mundo externo, simultáneamente construyen conocimiento al interior de sus mentes. Este nuevo conocimiento entonces les permite construir cosas mucho más sofisticadas en el mundo externo, lo que genera más conocimiento, y así sucesivamente en un ciclo autoreforzante.

Seymour Papert (1987) da el ejemplo de los ”engranajes” que tanto lo fascinaron en su infancia para plantear cómo pensar en un ”objeto con el cual pensar” y ”pensarse como un engranaje” le permitió aprender los conceptos formales matemáticos implicados en el desarrollo de un sistema de engranajes (relación entre dientes y fuerza en un tren de engranajes por ejemplo).

La propuesta central del construccionismo radica en plantear actividades de confección y construcción de artefactos aunque no necesariamente artefactos físicos. Estos, funcionan como facilitadores del aprendizaje porque los sujetos, al construir un dispositivo (un robot, un castillo o un programa de computadora), construyen también sus propias estructuras de conocimiento. Papert afirma que los sujetos además, deben construir objetos que sean de su interés personal, para poder interesarse mas en el proceso de construcción (y por ende, en su proceso de

construcción de conocimiento). Al mismo tiempo, los objetos construidos ofrecen la posibilidad de hacer más concretos, palpables y visibles (dentro del esquema de pensamiento del sujeto) los conceptos abstractos o teóricos, y por tanto, hacerlos más fácilmente comprensibles.

1.3. Ciencias de la computación

Tomando como referencia a la propuesta de la fundación Sadosky (2013) podemos hacer una diferencia entre las tecnologías de la información y comunicación (TIC) y las ciencias de la computación (CC), donde las primeras (TICs) se refieren a nivel global sobre el uso de herramientas basadas en computadoras (ofimáticas, navegación por internet por ejemplo), mientras que las ciencias de la computación están orientadas específicamente a la generación de habilidades y competencias especificas

1.4. Lenguajes formales

Un lenguaje formal es un lenguaje cuyos símbolos (alfabeto) y reglas para unir esos símbolos (gramática) están formalmente definidos (Giró, 2015).

Un lenguaje de programación es un lenguaje formal, como la lógica y la matemática, con la diferencia que fue diseñado para realizar procesos que pueden ser llevados a cabo por computadoras. Entre sus características principales, se encuentra la posibilidad de poder crear programas que gobiernen el comportamiento de una computadora, tanto a nivel físico (hardware) como lógico (software).

Durante la decada del cincuenta, en la primera época de las computadoras (las grandes mainframes como la ENIAC), estas eran programadas directamente escribiendo ”ceros y unos” sobre su memoria, utilizando lo que se define como

lenguaje de máquina, o una serie de reglas nmemo-técnicas llamadas **lenguaje ensamblador**. La dificultad de programar software cada vez más complejo, hizo que se desarrollaran lenguajes de programación cada vez más alejados del lenguaje de maquina y mas cercanos al lenguaje natural. Se dice comúnmente que un lenguaje de bajo nivel es aquel que se asemeja a la forma de trabajar de una computadora (assembler, por ejemplo), y que un lenguaje de alto nivel trata de ser mas parecido al lenguaje que usamos los seres humanos (python, por ejemplo); por lo tanto, es más fácil de entender para los programadores y para impartir cursos de iniciación a la programación (de Sevilla Vellón y Díaz, 2016).

1.4.1. Python

Python es un lenguaje de programación interpretado, de alto nivel y multi paradigma, cuya filosofía hace hincapié en una sintaxis que favorezca un código fácil de leer para los programadores.Como dice Marzal y cols. (2003), python es un lenguaje ideal para la enseñanza de programación porque tiene una *sintaxis* que facilita la economía de símbolos auxiliares (como el símbolo ";" usado para indicar el final de linea en lenguaje C), es *expresivo* (entendiendo esto como su capacidad para decir mucho con pocas lineas de código), y su *semántica* es elegante, haciendo que sea fácil de entender y escribir.

Figura 1.2: Logotipo de python

Entre sus características (Marzal y Luengo, 2004, pág 15) están:

- Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos.

- Python es muy legible. La sintaxis de Python es muy elegante y de fácil lectura.
- Python ofrece un entorno interactivo que facilita la realización de pruebas.
- Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.
- Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo.

1.5. Soberanía tecnológica

El concepto de soberanía tecnológica plantea una alternativa al proceso calificado por James Boyle (2003) como el segundo cercamiento de los bienes comunes. Si la acumulación inicial del capital se produjo mediante el cerco (enclosure) de las tierras comunales, este segundo cerco a los bienes comunes pretende la apropiación privada, mediante el sistema de marcas registradas, patentes y las leyes de propiedad intelectual, de objetos e ideas que hasta ahora quedaban excluidos por considerarse bienes comunes inapropiables. En este contexto, el uso de software libre sirve como herramienta que puede ser utilizada -como se propone en el presente proyecto- para lograr, siguiendo a Paulo Freire (2006), una pedagogía de la autonomía. Y no solo una base de usuarios cautivos para las empresas desarrolladoras de software.

1.6. Software libre

El software libre es un movimiento que comenzó en el año 1983 cuando Richard Stallman (2007) anunció el proyecto GNU en contra posición a la aparición

de monopolios artificiales en el desarrollo de software (Beatriz Busaniche, 2010). Se podría decir que la meta del movimiento fue dar libertad a los usuarios de programas de computadoras remplazando el software con términos de licencias restrictivas (software privativo) por una alternativa libre.

La comunidad de desarrolladores de software libre plantea que el software, para ser considerado libre, debe poder ser copiado, estudiado, distribuido y modificado libremente por cualquier persona o comunidad. En este sentido, se vuelve de vital importancia contar con el código fuente (y no sólo el código maquina o binario) de los programas para poder estudiarlos y modificarlos; además de tener una licencia (la licencia GNU/GPL) que proteja el derecho de autor y permita que ese código pueda ser distribuido sin el peligro de que sea apropiado por alguien más.

Por lo tanto para poder entender el concepto del software libre y las implicancias políticas del movimiento, lo principal es analizar los conceptos técnicos del desarrollo de software y la diferencia entre el código binario (archivo ejecutable) y el código fuente (escrito en un lenguaje de programación).

1.6.1. Código fuente

El software que diariamente usamos en nuestras computadoras esta compuesto por archivos binarios, largas listas de ceros y unos (código binario) con los cuales la computadora lee y ejecuta las instrucciones que estos archivos les brindan. Originalmente, en el desarrollo de software, los ingenieros escribían directamente sobre la memoria de las computadoras de la época, grandes mainsframes que ocupaban habitaciones completas. Con el crecimiento de la potencia de las computadoras, y la consiguiente complejidad en el desarrollo los programas necesarios para controlar estas nuevas máquinas, se comenzó a ver la necesidad de contar con una forma más eficiente y sencilla de crear dichos programas. Es en esta épo-

ca que nace la idea un programa compilador que convierta la información escrita en un lenguaje formal (con cierto grado de aproximación a un lenguaje natural humano) y la pase a un lenguaje de máquina (código binario).

Un compilador es, en su definición mas genérica, un programa que toma como entrada un texto escrito en cierto lenguaje y produce como salida otro texto en un lenguaje diferente (Grune y Catalina Gallego, 2007), manteniendo el archivo original (código fuente) y el archivo de salida (código objeto, generalmente un archivo binario ejecutable por la computadora). Es decir que un compilador convierte o traduce (como término más general) un archivo de código fuente a un archivo de código binario, permitiendo escribir en un lenguaje más parecido al lenguaje natural (y, por tanto, más sencillo de entender para los humanos).

Por consiguiente, podemos considerar al lenguaje de programación como un lenguaje formal, donde una serie de instrucciones inequívocas conforman un algoritmo (Giró, 2015) que puede ser convertido por un compilador en un archivo de instrucciones binarias para ser ejecutado por una computadora.

Richard Stallman (2007) realiza la analogía entre el código fuente y una receta de cocina (los pasos para hacer una torta por ejemplo), donde siguiendo instrucciones sencillas y no ambiguas se obtiene un producto final (la torta en este caso). En este esquema, la receta de cocina sería igual al código fuente, y el producto final al código binario (o ejecutable). Naturalmente, si nosotros sólo tenemos la torta, descubrir como obtener de vuelta la receta (ingeniería inversa) es mucho más difícil que en sentido inverso (receta-torta).

Para ejemplificar concretamente, en la siguiente tabla podemos ver las diferencias entre el software y el código fuente propuesta por Hart (2003).

Como plantean Jordi Adell y Bernabé (2007), cambiar un programa escrito en python, por ejemplo, para que en lugar de decir Hello World diga Hola Mundo sería bastante trivial y fácil de hacer. Sin embargo, hacerlo desde código binario se

Lenguaje de programación	Código fuente

ANSI C	`#include <stdio.h>`
	`int main(int argc, char* argv[])`
	`{`
	`puts("Hola mundo!");`
	`}`

Python	`print "hello world"` `exit()`
binario (”hello world” en ASCII)	01001000011001010110110001 10110001101111001000000101 01110110111101110010011011 0001100100

Cuadro 1.1: Lenguajes de programación, comparación entre código fuente y código máquina

vuelve muy complejo, y eso que solamente son los caracteres de la palabra Hello World y no un programa completo, el cual hasta el más sencillo, puede contener miles y hasta millones de ceros y unos.

Por tanto, para un programador es necesario contar con el código fuente de un software para poder modificarlo, estudiarlo y comprender su funcionamiento (y que no sea solamente una caja negra).

Teniendo en cuenta los conceptos mencionados, podemos explicar algunas de las definiciones que se utilizan para que un software sea considerado como software libre. Richard Stallman (2007) define cuatro libertades que tiene que tener el software para considerarlo libre:

- libertad 0: La libertad de usar el programa, con cualquier propósito (Uso)
- libertad 1: La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a las propias necesidades (Estudio).
- libertad 2: La libertad de distribuir copias del programa, con lo cual se puede

ayudar a otros usuarios (Distribución).

- libertad 3: La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie (Mejora).

Un programa es software libre si otorga a los usuarios todas estas libertades de manera adecuada. Por lo tanto, el echo de contar con el código fuente es una condición necesaria para poder ejercer las cuatro libertades que plantea la FSF (Free Software Foundation). Lo dicho hasta aquí supone que todos los programas desarrollados y distribuidos bajo licencias libres (por ej. -la licencia GNU/GPL V3) tienen que se distribuidos con los archivos de código fuente además de de los archivos ejecutables (en el caso de programas compilados).

El software libre busca proteger las libertades individuales de los usuarios de computadoras como oposición a los desarrollos de software privativos (como los define la FSF), licencias restrictivas de uso, software malicioso (malwares, backtrack) monopolios (Beatriz Busaniche, 2010), y prácticas poco éticas que algunas empresas pueden aplicar a sus usuarios. En cambio, gracias al software libre, los usuarios no están restringidos por el desarrollador de la aplicación y son dueños completos del programa que necesitan usar, permitiendo a comunidades, organismos estatales y/o universidades, adaptar dicho software a las necesidades concretas de cada grupo, y así poder dar respuesta a necesidades puntuales que podrían no ser ”rentables” para una empresa particular.

Figura 1.3: Logotipo del Free Software Foundation

El software libre es un desarrollo soportado por las comunidades y para las comunidades, donde el esquema de desarrollo es distribuido y global, o como dice (Raymond, 1998) un esquema dónde el desarrollo de software libre es más

parecido a un ”bazar” dónde todos aportan de manera ”desorganizada” y des centralizada, en contra posición a un desarrollo más parecido a una ”catedral”, dónde un arquitecto es el jefe central del desarrollo, en un esquema fuertemente estructurado.

1.7. Hardware de especificaciones abiertas

El movimiento de hardware libre (o hardware de especificaciones abiertas) , busca llevar el concepto del software libre (la libertad de usar, estudiar, distribuir o mejorar el software) al diseño de componentes físicos, especificando una licencia que permite distribuir planos y código fuente de desarrollos de PCBs (Printed Circuit Board por sus siglas en ingles) y hardware electrónico. Asimismo, se considera que un diseño de circuito (esquemático, diseño de PCB y archivos GERBER) debe ser desarrollado con software libre y usando formatos abiertos.

Figura 1.4: Logotipo del Open Source Hardware

De acuerdo a la declaración de principios de la **Open Source Hardware association** [1] , podemos decir que:

Hardware de Fuentes Abiertas (OSHW en inglés) es aquel hardware cuyo diseño se hace disponible públicamente para que cualquier persona lo pueda estudiar, modificar, distribuir, materializar y vender, tanto el original como otros objetos basados en ese diseño. Las fuentes del hardware (entendidas como los

[1] http://www.oshwa.org/definition/spanish/

ficheros fuente) habrán de estar disponibles en un formato apropiado para poder realizar modificaciones sobre ellas.

Idealmente, el hardware de fuentes abiertas debería ser diseñado de modo tal que puedan utilizarse en su construcción componentes, materiales y herramientas de alta disponibilidad y fácil acceso (en lo posible), empleando herramientas de fuentes abiertas (en el caso del software de desarrollo); esto permite maximizar la posibilidad de construir y usar ese hardware por parte de los usuarios.

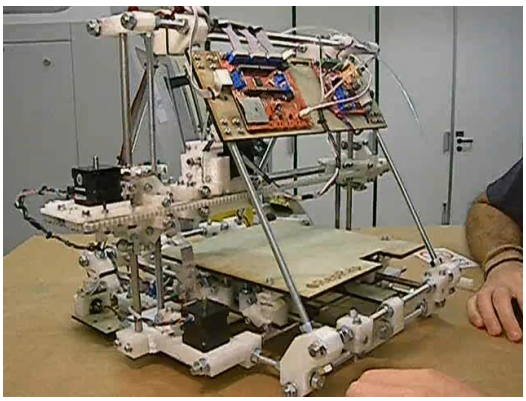


Figura 1.5: Impresora 3d diseñada con hardware libre

Es importante notar como la OSHW plantea que para que un hardware sea considerado de estándares abiertos, no sólo debe distribuirse sus planos y esquemáticos con un formato libre. Ademas, el diseño deberia contemplar la posibilidad de fabricar ese mismo hardware por parte de los usuarios. El hardware de fuentes abiertas da libertad de controlar la tecnología y al mismo tiempo permite compartir conocimientos.

González y cols. (2003) plantean una clasificación del hardware en función de su diseño y el software empleado para su creación, definiendo una clasificación primaria de los tres tipos de archivos necesarios para la fabricación de un PCB, el *esquemático*,el archivo de *pcb* y el archivo *GERBER*. En función de esa clasificación, se puede separar al hardware en tres tipos:

- (P): Software de diseño propietario.
- (L): Software de diseño libre.

- (M): Software de diseño propietario pero multi plataforma (funciona en sistemas operativos libres y también en propietarios)

Por tanto los tres archivos de construcción para el circuito electrónico, pueden ser de clasificación como enteramente libres (tipo **LLL**) o completamente propietarios (**PPP**), y todas sus combinaciones posibles.

1.7.1. Software libre en la escuela

Adell y Bernabé (2007) consideran al Software libre una alternativa para aplicar en el contexto del aula por sus ventajas pragmáticas (menor o hasta incluso nulo costo por licencias) que permiten ahorrar presupuesto, y por sus valores ético, políticos y sociales (Hart, 2003), que funcionan como disparadores de discusiones sobre los valores que una institución educativa tendría que promover.El software libre en la educación trata sobre la libertad de los docentes y dicentes, porque Como dice Paulo Freire (2015): *"Nadie libera a nadie, ni nadie se libera solo. Los hombres se liberan en comunión"*.

La incorporación del software libre en el desarrollo curricular del aula, promueve la cooperación entre las personas; si el software privativo la convierte en un delito (Adell y Bernabé, 2007), el software libre permite a las instituciones escolares sumar sus esfuerzos académicos a un proyecto global, independiente de los vaivenes económicos de las grandes corporaciones, y adaptable a las necesidades concretas de la comunidad donde ese establecimiento está asentado.

Usando software libre, los alumnos pueden disponer de copias gratuitas de los programas que necesiten para el trabajo escolar, sin restricciones de licencias que obligan a tener un software de menor calidad para presionar a los usuarios a comprar versiones "completas". Al disponer del código fuente, el mismo puede adaptarse a las necesidades del docente para situaciones concretas, como traducir

dicho software al idioma de sus alumnos (como el caso de la traducción de la suite ofimática Libreoffice al idioma aimara[2]).

Los proyectos de software libre suelen tener un coste inicial de desarrollo muy bajo (Adell y Bernabé, 2007). Generalmente empiezan como un desarrollo personal de algún programador o pequeño grupo de entusiastas y, gracias al trabajo global y distribuido logran crecer y obtener una ”masa critica” de desarrolladores. Como menciona Boyle (2003):

en una red global hay tanta gente y los costos son tan bajos que incluso los proyectos relativamente complejos atraen a las personas motivadas y capaces cuyo precio base ya ha sido superado

De esa forma, proyectos muy complejos pueden ver la luz, y ese mecanismo puede ser aprovechado por los colegios para ser generadores de contenido que pueda ser aprovechado a su vez por otras instituciones.

1.8. Robots

A nivel histórico, la palabra robots ha sido definida por la obra de teatro R.U.R. (Robots Universales Rossum) del dramaturgo checo Karel apek; allí se usó por primera vez la expresión robotnik"para referirse a seres humanos sintéticos creados para ser esclavos de la humanidad (Zabala, 2007). Sin embargo la popularidad del término robot se debe al escritor y divulgador científico Isaac Asimov, quien usó el termino robótica para designar a la disciplina que estudia a sistemas autónomos con cierto grado de capacidad para tomar decisiones e interaccionar con su medio (físico o virtual).

[2]http://www.elmundo.es/navegante/2007/08/03/tecnologia/1186167876.html

Se podría decir que cualquier sistema que posea sensores, actuadores y algún tipo de dispositivo que le permita realizar algoritmos de procesamientos, es un robot. Una definición tan laxa haría que prácticamente cualquier dispositivo pueda ser considerado un robot. Hay múltiples definiciones de la palabra robot, en función de las necesidades específicas de cada país u organización. Por ejemplo si tomamos la definición empleada por la R.I.A [3], un robot (sobre todo pensando en el robot industrial) es[4] :

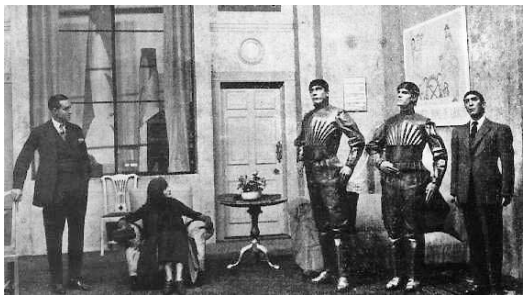


Figura 1.6: una escena de la obra de Karel Capek's R.U.R. (Rossum's Universal Robots), en donde podemos ver a los "robots".

un robot es un manipulador multifuncional y reprogramable diseñado para desplazar materiales, componentes, herramientas o dispositivos especializados por medio de movimientos programados variables con el fin de realizar tareas diversas. [Traducción propia]

Por otro lado, la J.A.R.A[5] (Japan Robot Association, ex J.I.R.A) usa una definición menos orientada a los manipuladores industriales (Reyes Cortés, 2011):

Los robots son dispositivos capaces de moverse de modo flexible análogo al que poseen los organismos vivos, con o sin funciones intelectuales, permitiendo operaciones en respuesta a las órdenes humanas

Como podemos ver, hay múltiples definiciones de la robótica, algunas más

[3] https://www.robotics.org/
[4] https://definitions.uslegal.com/r/robotics/
[5] http://www.jara.jp/e/index.html

orientadas a la industria, y otras más generales, donde un robot puede ser desde una androide (robot con forma humanoide) hasta un electrodoméstico.

1.9. Robótica educativa

El construccionismo de Papert promueve la utilización de computadoras como recurso para coadyuvar al desarrollo de nuevas maneras de pensar y aprender. Papert plantea que la única habilidad competitiva a largo plazo es la habilidad para aprender; su enfoque metodológico está orientado a usar las computadoras como herramientas para posibilitar ese desarrollo cognitivo en los alumnos. SÁnchez y Guzmán (2012) afirman que el construccionismo es reconocido como una teoría educativa que fundamenta el uso de la tecnología digital en educación.

En cuanto al uso de robótica como recurso educativo, SÁnchez y Guzmán (2012) resaltan que, por su carácter multidisciplinario, la robótica es una herramienta interesante como recurso facilitador del aprendizaje y el desarrollo de competencias generales. Permite trabajar transversalmente múltiples disciplinas, y sirve como motivador para que los discentes lleven a cabo proyectos mediante los cuales puedan experimentar y desarrollar sus actitudes cognitivas. En este sentido, y siguiendo a Pittí y cols. (2010), la robótica es una herramienta construccionista.

Uno de los mayores problemas de la implementación de la robótica en el aula, es la gran carga de contenido técnico que debe afrontar el docente para poder trabajar con su currícula. En consecuencia, varios fabricantes diseñaron Kits para el uso de robótica con fines educativos, sin embargo estos kits suelen ser extremadamente caros y por lo tanto restrictivos para su utilización masiva en las escuelas. El coste operativo de implementar un kit de robótica comercial puede ser prohibitivo para colegios de bajos ingresos, pero también el costo de mantenimiento (reparación y remplazo de piezas defectuosas, actualizaciones etc.) termina sien-

do un factor clave a la hora de usar los robots como herramientas pedagógicas, a causa del ”peligro” de que los alumnos rompan los robots y que reponerlos salga tan caro como comprar un kit nuevo.

La Robótica educativa con software y hardware libre es una opción más viable para su aplicación en el proceso de aprendizaje, porque permite a las escuelas adaptar la tecnología a las necesidades especificas de la institución, permitiendo reutilizar componentes que se encuentran en el colegio (aportados por la comunidad escolar), reciclando y ahorrando costos. A su vez por la gran expectativa que genera en los alumnos (y docentes), y al ser de código fuente libre, permite romper barreras culturales y políticas que pueden ir en detrimento de la calidad educativa; barreras como el alto costo de adquisición de los elementos, las licencias privativas para el software de control o la falta de documentación específica para comunidades minoritarias (traducciones a idiomas que no son consideraos viables comercialmente por ejemplo). La robótica educativa con software y hardware libre permite aprovechar las ventajas que ofrece la filosofía de desarrollo que promueven las comunidades de software libre, donde ”mil ojos ven más que uno” (Raymond, 1998), permite así involucrar a la comunidad escolar en su conjunto y que las soluciones aportadas por el grupo puedan ser utilizadas en otros colegios y vice-versa, generando una situaciones en las que todos los involucrados salen favorecidos.

Capítulo 2

Estado de la cuestión

2.1. Introducción

En este capítulo, se tratará de dar cuenta de la importancia de la la enseñanza de las ciencias de computación en la escuela media, señalando cómo la robótica puede servir de herramienta para lograr ese cometido, aprovechando sus particularidades y ventajas a la hora de abordar conceptos propios del pensamiento algorítmico (repeticiones, recursividad, saltos condicionales). Estos planteos se encuentran en consecuencia con la actual ley Nacional de Educación N°26.206 que, en su artículo 30°, Inc. F, enuncia:

Desarrollar las capacidades necesarias para la comprensión y utilización inteligente y crítica de los nuevos lenguajes producidos en el campo de las tecnologías de la información y la comunicación.

Podemos decir que las ciencias de la computación (Sadosky, 2013, pág 4) se volverán de vital importancia a la hora de pensar un esquema curricular especifico para abordar los contenidos básicos propuestos por dicha ley.

2.2. Antecedentes

2.2.1. Lenguaje LOGO

El uso de dispositivos robots para enseñanza, tiene su origen en el ya referido trabajo de Seymour Papert, y su desarrollo del lenguaje LOGO. Este lenguaje formal (Giró, 2015) fue diseñado para trabajar conceptos de matemáticas y geometría mediante la interacción con un robot con forma de tortuga (de ahí que el logotipo de LOGO sea una tortuga). Este dispositivo se movía sobre una superficie plana dibujando en función de las instrucciones previamente creadas en una computadora. Basado en ese esquema, los discentes programaban en lenguaje LOGO (un lenguaje muy parecido en su forma al lenguaje LISP), luego activaban el robot tortuga y la computadora enviaba las órdenes para que éste se mueva, dibujando sobre un papel.

A través de esa experiencia, Papert colabora con la empresa LEGO para fabricar el producto LEGO/LOGO, el cual después pasaría a ser conocido como LEGO MINDSTORM™, una plataforma física basada en fichas LEGO® que también incluye hardware de adquisición de datos capaz de leer sensores y trabajar sobre diversos actuadores.

Con el tiempo, el lenguaje LOGO abandonaría la tortuga robot. El elevado costo de cada robot tortuga y el abaratamiento de las micro computadoras (por ej. COMMODORE® c64) con capacidad de procesar gráficos a color hicieron que fuera muy difícil implementar una currícula basada en LOGO solamente usando elrobot tortuga.

El lenguaje LOGO está diseñado para ser sencillo de aprender, con instrucciones ”primitivas” (en el argot LOGO) que son intuitivas para el usuario; la palabra ’adelante’, hace que la tortuga avance N pasos, al igual que ”atrás” hace que retroceda. Sin embargo sigue siendo un lenguaje de programación completo y capaz

de hacer lo mismo que otros lenguajes de propósito general (como C, FORTRAN u otros lenguajes de la época).

Una de las ventajas de LOGO es su facilidad de poder hacer ”algo” con muy pocas lineas de código (típicamente algún dibujo con la ”tortuguita”), Seymour Papert (1987) plantea que cualquier niño, bajo las condiciones adecuadas, puede aprender un lenguaje de programación como LOGO.

Figura 2.1: robot ’tortuga’ original diseñado por el M.I.T

Originalmente, las primeras pruebas de LOGO se hicieron con robots como el de la figura 2.1, el cual estaba pensado para ser robusto, y capaz de soportar hasta el peso de un niño encima. Las computadoras de la época no tenían capacidad de procesar gráficos de alta definición (y las impresoras eran caras y de uso casi exclusivo en las empresas), por lo tanto la ventaja de disponer de un robot que dibuje eran muy atractivas.

Con el advenimiento de las micro computadoras (como las ZX spectrum™ y las commodores™ C64), que contaban con la capacidad de trabajar en monitores a color (o en los televisores CRT de la época) tenian bajo costo de adquisición, se dejó de utilizar a los robots tortuga, para directamente trabajar con el lenguaje de programación LOGO y una ”tortuga virtual”, menos llamativa pero mucho mas barata de adquirir para los colegios.

2.2.2. ARDUINO™ y la revolución del hardware libre.

El primer modelo de la placa Arduino™ fue introducido en 2005, ofreciendo un bajo costo y facilidad de uso para novatos y profesionales. Buscaba desarrollar proyectos interactivos con su entorno mediante el uso de actuadores y sensores. Su bajo costo y la enorme cantidad de documentación generada por las distintas comunidades de desarrolladores y entusiastas permitió que la plataforma Arduino™ se volviera un estándar a la hora de hablar sobre automatización o IOT (Internet of things).

Actualmente los miles de kits para enseñanza de robótica que se diseñan y fabrican (a pequeña o gran escala) están basados en la arquitectura Arduino™ y la serie de micro controladores AVR® de 8 bits. La inmensa documentación disponible y la facilidad de adquisición de los componentes, ha expandido su uso por parte de diversas comunidades y con diversas finalidades, entre ellas, la enseñanza de robótica. Esta gran demanda, también incidió en el abaratamiento de los costos de fabricación del hardware. Los kits generalmente constan de una serie uniforme de componentes electrónicos y mecánicos, como servo motores, motores de corriente continua, sensores analógicos / digitales y componentes de electrónica discreta (resistencias, leds, capacitores, entro otros), que permiten trabajar una serie de sistemas de automatización, robótica y/o domótica de mayor o menor complejidad.

Figura 2.2: Versión 'mini' del robot tortuga controlado por un micro computador Apple

2.3. lenguajes de programación en la enseñanza

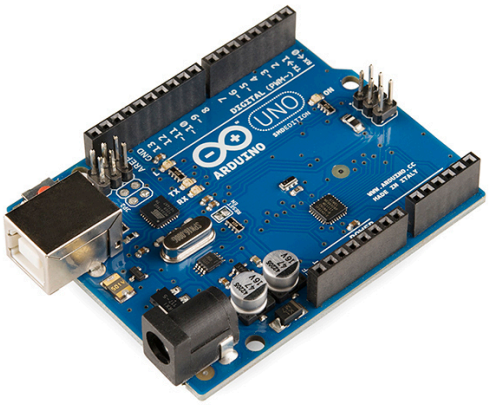


Figura 2.3: Arduino™ Uno Rev. 3

En la actualidad y en el marco de la ley Nacional de educación, la enseñanza de la ciencia computacional está tomando relevancia a nivel curricular. La falta de profesionales especializados en programación y la constante demanda de ”mano de obra” por parte de la industria del software (software factory) se ha vuelto un factor de preocupación por parte de las autoridades nacionales. Para tratar de paliar esa problemática, se están implementando distintos planes y programas gubernamentales. Dentro de estos proyectos podemos encontrar el Programa **Escuelas del Futuro**, situado en la órbita de la Secretaría de Innovación y Calidad Educativa del Ministerio de Educación y Deportes de la Nación. El mismo busca:

crear un Proyecto con el objeto de generar un cambio transformador en las estrategias pedagógicas y políticas de contenidos para integración del sistema educativo a la cultura digital. Que para el logro de los objetivos previstos en los considerandos precedentes, resulta conveniente la creación del Proyecto ”Escuelas del Futuro” orientado a propiciar alfabetización digital de todos/as los/as estudiantes de la Argentina, a través de integración de áreas de conocimiento emergentes, como la programación y la robótica. [1]

[1]Resolución Ministerial 2.376/16 Ministerio de Educación y Deportes, consultado en

Desde esa perspectiva, y como señalan desde la Fundación Sadosky, la alfabetización digital parte de la enseñanza de la ciencia computacional pensada como un conjunto amplio de fundamentos y principios independiente de tecnologías (Sadosky, 2013, pág 12) que incluyen:

1. Programación y algoritmos
2. Estructuras de datos
3. Arquitecturas y redes de computadoras

La ciencia computacional permite fomentar habilidades que pueden ser aplicadas a variados campos de estudio como:

1. Modelización y formalización.
2. Descomposición en sub problemas.
3. Generalización y abstracción de casos particulares.
4. Proceso de diseño, implementación y prueba.

De esta forma la enseñanza de un lenguaje formal toma una particular relevancia dentro del esquema educativo actual, que busca resolver los problemas planteados por la necesidad de alfabetizar digitalmente a la población.

Sin embargo, hay críticas concretas a la enseñanza del ”pensamiento algoritmico” en contraposición a la enseñanza de lenguajes formales específicos. La idea de pensamiento algorítmico es que se pueden aprender conceptos de programación a través de meta-lenguajes, o lenguajes pedagógicos (pseudo código o sistemas basados en diagramas y gráficos). Desde esta perspectiva, un lenguaje

http://www.saij.gob.ar/proyecto-escuelas-futuro-nv15957-2016-12-05/123456789-0abc-759-51ti-lpssedadevon?

educativo permitiría simplificar el contenido técnico complejo, y abordar de forma escalonada el aprendizaje de la ciencia computacional.

Como critica a ese modelo, Dijkstra y Montoya (2010) plantean que la ”simplificación” de un lenguaje formal con pseudo código o con lenguajes ”de juguete” (como el caso del lenguaje BASIC en su momento), puede ser contraproducentes para el aprendizaje. En terminaos del propio Dijkstra:

Es prácticamente imposible enseñar una buena programación a los estudiantes que han tenido una exposición previa a BASIC: como programadores potenciales son mentalmente mutilados más allá de la esperanza de regeneración.

[Traducción propia]

Para Seymour Papert (1987) los lenguajes simplificados como BASIC (o como los lenguajes basados en bloques gráficos de mas reciente aparición) que se ”promocionan” como lenguajes simples de aprender a causa de su reducido vocabulario, son sin embargo, extremadamente complejos de usar para crear programas no triviales. El fenómeno del lenguaje BASIC fue definido por Seymour Papert (1987) como el ”fenómeno QWERTY”, a causa de los teclados, que fueron diseñados en la época de las primeras máquinas de escribir mecánicas. Estas máquinas solían trabarse si se ponían cerca las teclas de uso más común; por ello, se diseño un sistema de ordenamiento de letras denominado QWERTY que dificultaba la digitación veloz para evitar dicho problema. Aun después de la creación de las computadoras, cuando los problemas mecánicos de las maquinas de escribir ya no tenían sentido, se continuo usando ese mismo sistema. El el caso de los sistemas BASIC, sucedió algo similar: fueron diseñados para resolver problemas presentados por las primeras micro computadoras, simplificando su set de instrucciones. Una vez que esos problemas fueron resueltos por hardware más potente y de menor costo, la industria no obstante siguió eligiendo BASIC como herramienta de desarrollo y enseñanza de programación.

2.3.1. Competencias especificas para el aprendizaje de un lenguaje de programación

Para usar un lenguaje de programación (entendiendo este como un lenguaje formal de tipo especifico) como herramienta de enseñanza, es necesario tener en cuenta una serie de capacidades o características para que puedan ser incluidos en una currícula educativa. En general podemos decir que un lenguaje de programación debería tener las siguientes cualidades:

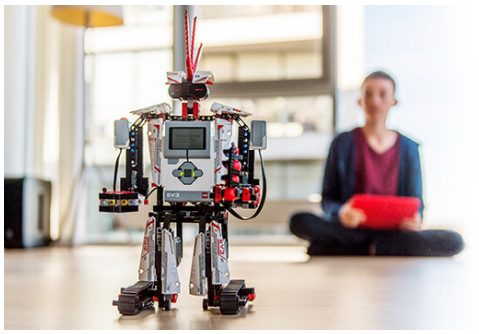


Figura 2.4: Robot LEGO® MINDSTORM®

1. Ser de alto nivel (similar a un lenguaje natural).

2. Ser un lenguaje de propósito general (la capacidad de crear cualquier programa).

3. Tener capacidad multi paradigma (programación orientada a objetos, programación estructurada, etc.).

4. Ser multi plataforma (capacidad de adaptarse a distintos dispositivos).

También, es importante que un lenguaje de programación seleccionado para trabajar en el aula sea de uso común y extendido en la industria, no tanto por su éxito comercial momentáneo, que puede variar en muy poco tiempo, si no por la capacidad de conseguir documentación especifica y comunidades de soporte que permitan desarrollar al máximo las capacidades intrínsecas de la herramienta.

Hay que hacer una aclaración respecto a los lenguajes de marcado (como HTML), que no son lenguajes de programación propiamente dicho dado que no tienen características básicas como:

- variables
- repeticiones
- sentencias condicionales
- recursividad

Si bien esta enumeración de características no es exhaustiva, pone en manifiesto que, habiendo una enorme cantidad de lenguajes de programación, algunos de características mas especificas que otros, la elección del lenguaje de programación que se usará para un curso de enseñanza inicial no es un asunto trivial.

2.4. Robótica y la enseñanza de programación

En la actualidad, el uso de robótica se está extendiendo dentro de los espacios curriculares, los robots educativos se presentan como una alternativa interesante para poder trabajar conocimientos transversales a distintas disciplinas, desde la matemática y programación hasta las ciencias naturales. No obstante, se podría decir que dónde más se pueden aprovechar las ventajas de los kits de robótica

que pululan en el mercado es precisamente en la enseñanza de los lenguajes de programación.

Básicamente un robot es una computadora con sensores y actuadores que le permiten interactuar en un entorno físico; para lograr ese cometido el robot debe ser programado con un algoritmo que le permita resolver las situaciones complejas que pueden ocurrir mientras se mueve por un medio ambiente físico. A diferencia de lo que ocurre en un entorno virtual como un simulador robótico, interactuar en un espacio físico obliga al diseñador del algoritmo a tomar medidas de corrección y control mediante sensores, y dotar al robot de cierta ”inteligencia” para resolver situaciones inesperadas, como fallas mecánicas o defectos de fabricación. Esto ultimo sobre todo cuando se usan componentes reciclados o de baja calidad.

Es importante hacer una distinción entre ”aprendizaje DE robótica ” y ”aprendizaje CON robótica” (Malec, 2001), entendiendo que la robótica es una disciplina en sí misma, y su uso en la industria tiene un nivel de complejidad que requiere un grado de especialización extra para los técnicos e ingenieros que trabajen en ese ámbito. Por eso mismo el aprendizaje CON robots, debería ser pensado como un medio para facilitar la construcción por parte de los discentes de conocimientos transversales adaptando las complejidades inherentes al diseño y construcción de un robot. Este aprendizaje CON promueve en los estudiantes el desarrollo de competencias tales como: la toma de decisiones basadas en el conocimiento, el formular explicaciones científicas y el trabajo en equipo (López Ramírez y Andrade Sosa, 2013).

Capítulo 3

Proyecto de transferencia

3.1. Introducción

ILa implementación del uso de robots como herramientas pedagógicas para la enseñanza de lenguajes de programación implica una serie de compromisos técnicos y logísticos específicos, desde la adquisición de herramientas para la fabricación y mantenimiento de los mismos, hasta la capacitación de los docentes para poder trabajar en el aula.

Dado que un robot es un componente físico, y por lo tanto tiene un coste de adquisición y mantenimiento que puede ser elevado, las decisiones de costo/beneficio son fundamentales -aunque no exclusivas- a la hora de optar utilizar la robótica en un colegio.

Este proyecto de transferencia busca generar la documentación necesaria para que las escuelas técnicas con orientación a la informática o electrónica puedan adoptar e implementar el proyecto de robótica educativa ICARO, poniendo énfasis en la capacitación de docentes y discentes para fabricar el hardware y usarlo en enseñanza de lenguajes de programación.

3.2. Justificación

La implementación de un proyecto de hardware libre, requiere capacitación especifica en el uso de herramientas de diseño CAD/CAM libres (Bareño, 2011), además de una migración de los laboratorios informáticos de la institución, para que puedan trabajar con sistemas operativos GNU/Linux (el sistema propuesto para este proyecto de transferencia).

Los proyectos de hardware libre buscan permitir a las instituciones, usuarios finales y hasta empresas independizarse de un proveedor especifico de tecnología (González y cols., 2003). De esa manera las instituciones escolares pueden optimizar su presupuesto en función de las necesidades especificas de cada instituto.

El proyecto pretende transferir efctivamente el ”saber como” (Know-how), para que la institución participante pueda re diseñar, fabricar e implementar placas NP07 del proyecto ICARO, contemplando desde la obtención y compra de los elementos de fabricación, hasta su implementación en el aula, así como la adecuación del laboratorio informático para el trabajo con el hardware propuesto.

3.3. objetivos

generales

1. Promover el uso de software y hardware libre en los colegios técnicos.

2. Transferir el conocimiento para implementar la fabricación de hardware del proyecto ICARO en la institución participante.

3. Abaratar costos para la institución por el uso de licencias libres.

4. capacitar a los docentes en la enseñanza de lenguajes de programación.

especificas

1. Preparar un espacio de la institución como laboratorio de electrónica.

2. Capacitar a docentes en la construcción del hardware propuesto.

3. Capacitar a los docentes en el uso de la propuesta curricular para el hardware NP07.

3.4. licencias

Todas la documentación del proyecto (código fuente, planos esquemáticos del hardware y documentación pedagógica) esta distribuida con las licencias del proyecto GNU[1] o de la fundación *creative commons*[2], la elección de una u otra forma de licenciamiento esta determinado de la siguiente forma (Bareño, 2011):

- En el caso de todo el código fuente (python o C), planos esquemáticos del hardware y planos del pcb, sera distribuido con la licencia GNU/GPL V3[3] [4].

- Para toda la documentación que no sea referida a los temas anteriores (documentación técnica) sera distribuida con el formato de licenciamiento de la fundación *creative commons*, usando el formato de licencia " Reconocimiento Compartir Igual (by-sa)[5]". Licencia que permite el uso comercial del proyecto y de las posibles obras derivadas del mismo, teniendo en cuenta que la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original del proyecto.

[1] https://fsf.org/es
[2] http://creativecommons.org.ar/
[3] https://lslspanish.github.io/translation_GPLv3_to_spanish/
[4] https://www.gnu.org/licenses/gpl3.0.en.html
[5] http://creativecommons.org.ar/licencias

3.5. metodologia

La metodología propuesta para el proyecto de transferencia se basa en tres etapas, una primera instancia donde se acondicionara la institución para poder implementar la fabricación del hardware para luego en la segunda etapa, capacitar a los docentes en la fabricación y mantenimiento de las placas NP07. La tercera etapa es la implementación dentro del aula

Luego de la primera instancia de presupuesto y adquisición de materiales, se procederá a impartir el curso de capacitación y acompañamiento a los docentes participantes del proyecto, donde se acondicionara un espacio de la institución para funcionar como laboratorio de desarrollo electrónico, ademas de taller para la fabricación del hardware del proyecto (las placas NP07). Durante la primera mitad del curso, los participantes se instruirán en el uso de alternativas libres para el desarrollo de PCBs mediante el uso de software CAD/CAM KICAD, así mismo se tomara contacto con cada modulo de fabricación del hardware poniendo el énfasis en el puntos de comprobación y reparación. La idea de este primer encuentro es poder comprender el funcionamiento y construcción de las placas.

La segunda mitad del curso de capacitación esta destinada a el desarrollo de las secuencias didácticas especificas que los docentes darán en los cursos de lenguajes de programación, usando el hardware NP07 como herramienta para el desarrollo de las competencias especificas de las ciencias de la computación (Sadosky, 2013).

La tercer instancia sera el seguimiento y ayuda de los docentes por parte del equipo de transferencia durante el dictado del curso de enseñanza del lenguaje de programación PYTHON a los discentes de la institución. El dictado de este curso dependerá de cada institución, si deciden ponerlo dentro de un espacio curricular o extra curricular, pero la metodología en ambos casos se basara en la construcción y programación de un robot usando componentes reciclados de equipos de

electrónica (impresoras, scanners, equipos reproductores de música).

Al final del curso, los docentes y dicentes podrán generar un documento de retro alimentación con sugerencias para mejorar el desarrollo del proyecto de transferencia tecnológica educativa y que pueda servir para futuras implementaciones en otras instituciones.

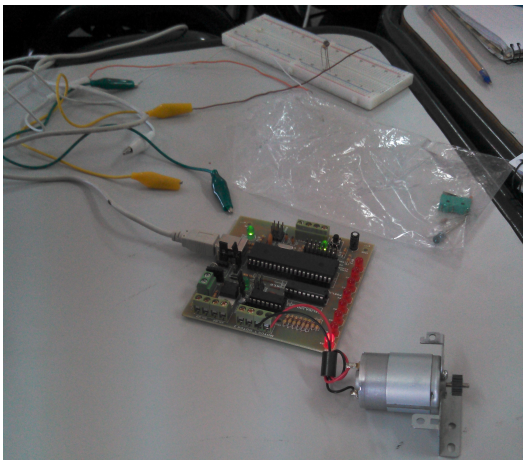


Figura 3.1: placa para robotica educativa NP07

3.6. Capacitación

El esquema de capacitación esta elaborado en los anexos 1, 2 y 3, donde el anexo 1 es el esquema de armado del hardware NP07 (guiá de construcción) diseñada por la comunidad de usuarios y desarrolladores de ICARO, el anexo 2 es la capacitación para la construcción del hardware NP07 para docentes y el anexo 3 son las secuencias didácticas diseñadas para trabajar con los discentes durante el segundo cuatrimestre.

Las clases serán programadas para ser cursadas 1 ves a la semana, durante el horario escolar, o en horario extra escolar, en función de la elección de la institución con respecto a la cursada.

3.7. duración

La duración del proyecto sera de 1 año escolar, donde el primer cuatrimestre escolar se tomara para el presupuestado de componentes y la capacitación docente para, luego en el segundo cuatrimestre, pasar al curso de enseñanza de programación con robots para los discentes de la institución.

Terminado el proyecto, la institución educativa contara con el apoyo de la comunidad de desarrollo de ICARO a través de los canales habituales para consultas y recomendaciones que se manejan en los proyectos de desarrollo de software libre (Salas de chat IRC, listas de correo y foros).

3.8. gantt

El diagrama de gantt propuesto, contempla un proceso de un año de trabajo dentro de la institución escolar que participara de el proyecto de transferencia tecnológico educativo, donde el primer cuatrimestre de año escolar se utilizara para poder capacitar a los docentes y preparar las instalaciones para poder impartir un curso inicial de robótica con los discentes. Al final del año escolar, los discentes podrán mostrar un trabajo final usando el hardware propuesto.

El diagrama esta propuesto para tener seis instancias de intervención:

1. **Análisis y evaluación de la institución**: En esta etapa, se determinara las necesidades del colegio, las características y estado de las instalaciones y en que etapa de la formación escolar se implementara la curricula de enseñanza de programación (tomando como referencia las practicas educativas que se vengan implementando en la escuela).

2. **Presupuesto:** Teniendo en cuenta las posibilidades económicas de la institución, se tomara las decisiones presupuestarias de equipamiento para el

laboratorio y la adquisición de componentes y herramientas para la fabricación de las placas NP07 del proyecto ICARO. En esta etapa se buscaran los proveedores zonales para componentes electrónicos

3. **Compra de componentes:** Aprobado el prepuesto, se procederá a comprar todas las herramientas y componentes necesarios para poder acondicionar la instalación destinada a alojar el laboratorio de desarrollo.

4. **Capacitación docente:** Con las instalaciones preparadas y acondicionadas, se trabajara con los docentes seleccionados en un curso de capacitación de dos meses de duración y cursada semanal (ocho clases en total). La idea principal es que los mismo docentes puedan fabricar las primeras placas NP07 para poder luego enseñarle a sus discentes como hacerlo, aprendiendo las características técnicas del hardware y el software de trabajo.

5. **Curso para los discentes de la institución:** Terminado el primer cuatrimestre, se implementara el curso de programación para los discentes seleccionados previamente, con la idea de poder trabajar con los conceptos básicos de las ciencias de la computación, a través del diseño y desarrollo de un robot educativo y la programación mediante lenguaje PYTHON del mismo.

6. **Trabajo final de los discentes:** En esta etapa, los discentes que participaron del curso, trabajaran en un proyecto de final de curso, donde diseñaran un robot y lo podran exponer en una feria de ciencia escolar que prepare la institución. Esta etapa se utilizara como parte de una auto evaluación del proyecto, para poder darle seguimiento el año siguiente y poder ”pulir” falencias y proponer mejoras.

La finalidad del proyecto es lograr que en un año escolar, la institución pueda

estar preparada para fabricar, reparar y mantener el hardware de el proyecto ICA-RO, para luego poder utilizarlo como herramienta pedagógica para los cursos de enseñanza de lenguajes de programación.

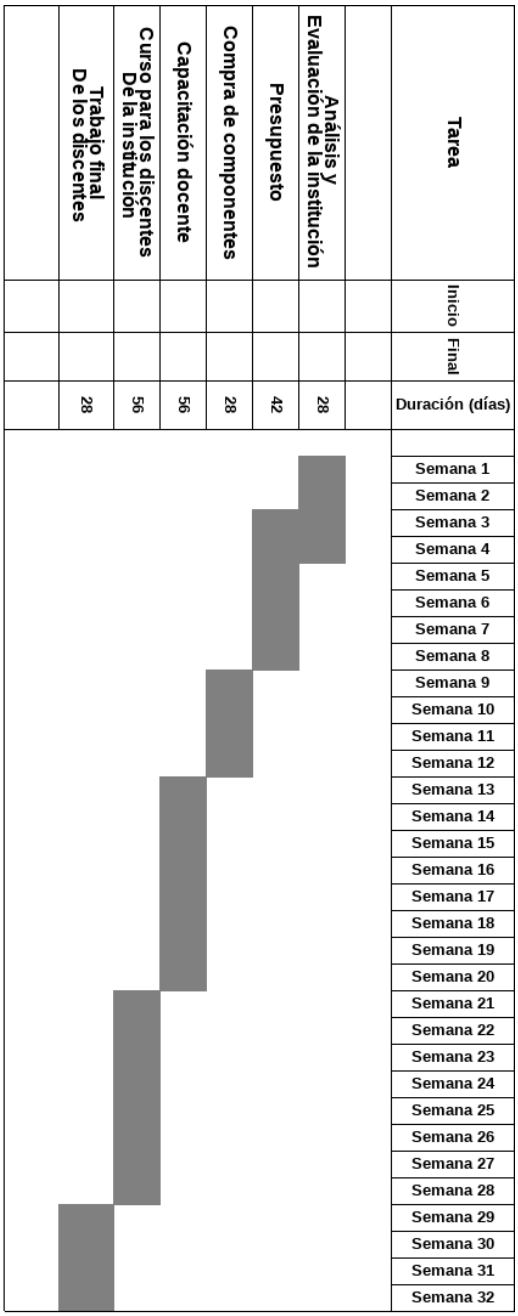


Figura 3.2: diagrama de gantt

3.9. requerimientos para el laboratorio de desarrollo

Para poder montar un laboratorio de desarrollo electrónico, la institución participante deberá acondicionar un espacio con las siguientes características:

- Mesas de trabajo amplias para poder disponer del equipamiento.
- Zapatillas eléctricas para disponer de los soldadores de estaño.
- Pizarrón y pantalla para proyector.
- Laboratorio de Computadoras (pueden ser Pcs de escritorio o notebooks) preparadas con sistema GNU/Linux.
- mueble tipo placar con cerradura, para almacenar las herramientas y componentes electrónicos.

En función de la cantidad de participantes que se determine en la etapa de evaluación y el presupuesto aprobado, se tendrán en cuenta las dimensiones de necesarias para el laboratorio así como las cantidades necesarias de los elementos requeridos para el desarrollo del proyecto. Como referencia se debe tener en cuenta que que para poder trabajar en el laboratorio, los discentes deberán formar grupos de entre 3 y 5 integrantes.

3.10. herramientas

Las herramientas necesarias para la fabricación del las placas NP07 se pueden dividir en 3 tipos:

1. Herramientas especificas del laboratorio y que solo es necesario adquirir una para el laboratorio.

2. Herramientas para la fabricación de las placas NP07 y que se repartirán en formato ”kit” por cada grupo de trabajo.

3. Herramientas generales para el diseño de un robot y trabajo en el aula.

Las herramientas especificas del laboratorio son todas las que permiten el trabajo dentro del mismo, pero que no forman parte especifica del armado del hardware y que pueden ser adquiridas en pequeñas cantidades, siendo herramientas auxiliares para la tarea. Los requerimientos de estas herramientas son:

- Equipo proyector de video.
- Programador electrónico para micro controladores PICs de la firma Microchip®.
- Juego de destornilladores para electrónica.
- Mini torno con set de mechas y accesorios.
- Perforadora de mano.
- Perforadora de banco.
- Impresora láser mono cromática.
- Plancha de mano (para la fabricación de PCBs caseros).

Algunas herramientas opcionales, que por su excesivo precio no son incluidas entre las requeridas:

- Impresora 3D.
- Osciloscopio digital de dos canales.
- Mini Fresadora CNC (Control numérico por computadora) para circuitos electrónicos.

- Perforadora de banco.

En el caso de las herramientas que se usaran para la fabricación de las placas NP07, se necesita comprar el siguiente Kit de herramientas básicas para soldadura con estaño. En función de la cantidad de participantes (si el curso de fabricación se hará extensivo a docentes y discentes) es recomendable comprar un Kit por cada 4 participantes, seria ideal adquirir entre 5 y 10 Kits de soldadura.

Los componentes del kit para soldadura son:

- Soldador de estaño tipo cautin de punta cerámica y 40W de potencia.
- Protoboard doble de 1360 puntos con terminales de alimentación y base metálica.
- Alicate de corte.
- Pinza de punta.
- De-soldador a pistón.
- Soporte multi propósito con lupa.
- Cinta malla de-soldante 1,5 Mts.
- Soporte soldador con esponja de limpieza.
- 1,5 Mts Estaño 60/40 0,8mm con fundente.
- Multimetro digital.

Las herramientas generales necesarias, sera usadas para fabricar robots, dar mantenimiento al equipamiento adquirido y para reciclar o re utilizar componentes electro-mecanicos que puedan aprovecharse del descarte de componentes en desuso que la institución consiga. Las herramientas basicas son:

- Set de destornilladores relojeros.
- Pinzas.
- Tijeras de cartón.
- Pistola encoladora de silicona.
- Cúter (trincheta).
- reglas metálicas.

3.11. laboratorio informatico

Teniendo en cuenta que el uso del robots esta planeado como un sistema auxiliar para la enseñanza de programación en los discentes, la instalación de un laboratorio informático deberá contar con las siguientes características a nivel de software:

- Sistema GNU/ Linux instalado (Fedora o Huayra linux).
- Entorno gráfico XFCE (recomendado por su poco consumo de memoria RAM).
- Interprete python 2.7 y python 3.0.
- Consola interactiva para python (IPYTHON).
- Entorno de desarrollo integrado (IDE) para python (GEANY).
- Software de desarrollo electronico (KICAD).
- Software de diseño 3d (BLENDER Y FreeCad).
- Software de diseño vectorial e imágenes (Gimp, INKSCAPE).

- Software de control del hardware (ICARO).
- Software para cargar bootloaders (PK2CMD).
- Editor de texto (LibreOffice).
- Librerías para tratamiento de imágenes con python (OPEN-CV).

EL laboratorio, al igual que con el kit de herramientas de soldadura, se recomienda que haya por lo menos 1 computadora cada 4 o 5 discentes. Con respecto a la instalación del sistema GNU/Linux, se pueden tomar 3 opciones:

1. Formatear las computadoras y solo instalar GNU/Linux (single boot).
2. Instalar GNU/Linux en una partición del disco, conservando el sistema operativo anterior (dual Boot).
3. Tener pen drives preparados con GNU/Linux y arrancar el sistema operativo desde el pen drive cada ves que se necesite usar la maquina (sistema LiveUsb Linux).

En función de las necesidades del colegio y de las especificaciones del laboratorio informático, se deberá tomar una de las 3 opciones disponibles.

Referencias

Adell, J., y Bernabé, Y. (2007). Software libre en educación. *Tecnología educativa. Madrid: McGraw-Hill*, 173–195.

Bareño, C. C. (2011). Metodología para la transferencia tecnológica en la industria electrónica basada en software libre y hardware copyleft. En *de xvii workshop de iberchip, bogotá, colombia.*

Beatriz Busaniche. (2010). *Argentina copyleft. La crisis del modelo de derecho de autor y las prácticas para democratizar la cultura.*

Boyle, J. (2003). The second enclosure movement and the construction of the public domain. *Law and contemporary problems*, *66*(1/2), 33–74. Descargado 2016-08-27, de `http://www.jstor.org/stable/20059171`

Chavarría, J. V. (2011, marzo). Software libre, alternativa tecnológica para la educación. *Revista Actualidades Investigativas en Educación*, *5*(2). Descargado 2016-08-26, de `http://revista.inie.ucr.ac.cr/index.php/aie/article/view/86` doi: 10.15517/aie.v5i2.9150

de Sevilla Vellón, M. A. F., y Díaz, M. J. A. (2016). Introducción práctica a la programación con python.

Dijkstra, E. W., y Montoya, É. S. (2010). Por qué johnny no puede comprender. *Lámpsakos*(3), 76–77.

Freire, P. (2006). *Pedagogía de la autonomía: saberes necesarios para la práctica educativa*. Siglo XXI. (Google-Books-ID: OYK4bZG6hxkC)

Freire, P. (2015). *Pedagogía del oprimido.* Buenos Aires: Siglo veintiuno. (OCLC: 946344968)

Giró, J. (2015). *Lenguaje formales y teoría de autómatas*. Buenos Aires: Alfaomega. (OCLC: 953193943)

González, I., González, J., y Gómez-Arribas, F. (2003). Hardware libre: clasificación y desarrollo de hardware reconfigurable en entornos GNU/Linux. En *VI Congreso de Hispalinux, Universidad Rey Juan Carlos I.*

Grune, D., y Catalina Gallego, A. (2007). *Diseño de compiladores modernos*. Madrid: McGraw-Hill. (OCLC: 182734548)

Hart, T. D. (2003). *Open source in education*. May.

López Ramírez, P. A., y Andrade Sosa, H. (2013). Aprendizaje con robótica, algunas experiencias. *Educación*, *37*(1).

Malec, J. (2001). Some thoughts on robotics for education. En *2001 aaai spring symposium on robotics and education.*

Marzal, A., Llorens, D., y Gracia, I. (2003). *Aprender a programar con python: una experiencia docente.* Universitat Jaume I, Obtenido de: http://es. tldp. org/Presentaciones/200309hispalinux/15/15. pdf.

Marzal, A., y Luengo, I. G. (2004). *Introducción a la programación con python y c*. Publicacions de la Universitat Jaume I.

Montoya, M. S. R., y Aguilar, J. V. B. (2012). Recursos educativos abiertos en ambientes enriquecidos con tecnología.

Papert, S. (1993). *The children's machine: rethinking school in the age of the computer*. New York: BasicBooks. (OCLC: 27264650)

Pittí, K., Curto Diego, M. B., y Moreno Rodilla, V. (2010). Experiencias construccionistas con robótica educativa en el Centro Internacional de Tecnologías Avanzadas. *Teoría de la Educación*, *11*(1), 26. Descargado 2016-08-26, de `http://gredos.usal.es/jspui/handle/10366/72852`

Raymond, E. S. (1998). La catedral y el bazar. *The Linux Logic Home Page*, *12*. Descargado 2016-08-26, de `http://www.athanazio.com/downloads/livros/catedral-ou-bazar.pdf`

Reyes Cortés, F. (2011). *Robótica: control de Robots manipuladores*. México: Alfaomega. (OCLC: 931981753)

Sadosky, F. (2013). Cc-2016 una propuesta para refundar la enseñanza de la computación de las escuelas argentinas. *Buenos Aires*.

SÁnchez, F. A. B., y Guzmán, A. F. (2012). La robótica como un recurso para facilitar el aprendizaje y desarrollo de competencias generales. *Education in the knowledge society (EKS)*, *13*(2), 120–136. Descargado 2016-08-21, de `https://dialnet.unirioja.es/servlet/articulo?codigo=3979056`

Seymour Papert. (1987). *Desafío de la mente*.

Stallman, R. M. (2007). *Software libre para una sociedad libre*. Traficantes de Sueños. (Google-Books-ID: ww9XAgAACAAJ)

Zabala, G. (2007). *Robótica*. Buenos Aires: Gradi.