UNIVERSIDAD NACIONAL DE LA PATAGONIA
SAN JUAN BOSCO

Facultad de Ingeniería

Título: Extensión del lenguaje de programación DaVinci-C e integración con la robótica educativa

---

Trabajo de tesis realizado como requisito para la obtención del título de Licenciado en Informática

Alumno: A.P.U. Jonatan Antonio Mamaní Ramirez

Directora: Lic. Beatriz Depetris

Co- Director: Lic. Daniel Aguil Mallea

Junio de 2018

Ushuaia, Tierra del Fuego, Argentina

Agradecimientos

Son muchas las personas a las que quisiera expresar mi más sincero agradecimiento, principalmente a mi familia y a mi compañera, Vanesa Caipichun, por apoyarme y animarme desde el primer día y hasta la actualidad. También, a mi directora de tesis, la Lic. Beatriz Depetris, y a mi codirector, el Lic. Daniel Aguil Mallea, por guiarme y orientarme en el desarrollo del proyecto, supervisarlo y corregirlo.

A mi compañero de trabajo, el Lic. Carlos Germán Tejero, por su permanente motivación, ayuda y apoyo necesario en el transcurso de estos últimos años.

A todos los que confiaron en mí y colaboraron, directa o indirectamente, para finalizar mis estudios universitarios.

Índice principal

Agradecimientos I
Índice principal II
Índice de tablas V
Índice de figuras VI
Capítulo 1 - Introducción 1
1.1 Objetivos 2
1.1.1 Objetivo general 2
1.1.2 Objetivos específicos 2
1.2 Alcance de la tesis 2
1.3 Lenguajes de programación - Breve síntesis 3
1.4 Enseñanza de la programación 4
1.4.1 Programación en los niveles iniciales 5
Capítulo 2 - DaVinci Concurrente 7
2.1 Características del actual desarrollo 7
2.2 Ventajas y desventajas 8
2.3 Características del lenguaje e intérprete DaVinci-C 8
2.4 Mejoras propuestas al lenguaje 8
2.4.1 Justificación de las mejoras propuestas: importancia del uso de las funciones 9
2.5 Implementación del subprograma tipo función en el DaVinci-C 10
2.5.1 Especificación en el lenguaje del tipo función 12
2.6 Funciones definidas por el lenguaje que se incorporan 14
2.6.1 Definición de las nuevas funciones primitivas del lenguaje 14
Capítulo 3 - Robótica educativa 18
3.1 Historia 18
3.2 Clasificación de los robots 19
3.2.1 Autonomía 21
3.2.2 Tipos 21
3.3 Arquitectura de los robots 21
3.4 Aplicaciones de la robótica 22

3.4.1 La robótica y su aplicación en la educación 23
3.5 Placas programables 24
3.5.1 Robots educativos programables 25
3.5.2 Niveles iniciales 29
3.5.3 ¿Qué es arduino? 30
3.5.4 ¿Qué es Lego Mindstorms? 31
Capítulo 4 - Integración de la robótica educativa en DaVinci-C 35
4.1 Problemas con los actuales lenguajes y entornos de desarrollo 36
4.2 Protocolo de comunicación entre el lenguaje y el robot 37
4.2.1 Palabras reservadas del lenguaje de programación DaVinci-C 38
4.2.2 Subconjunto de primitivas 38
4.2.3 Primitivas del robot 39
4.2.4 Protocolo 43
4.3 Elección de las herramientas utilizadas 44
4.4 Comunicación entre el intérprete y el robot 45
4.5 Recursos limitados 45
4.6 Especificación e implementación del intérprete 46
4.6.1 Requerimientos 46
4.6.2 Diagrama de clases 46
4.6.3 Definición de la implementación 47
4.7 Especificación de la ciudad física 48
4.7.1 Superficie 49
4.8 Lego Mindstorms EV3 50
4.8.1 Características principales 50
4.8.2 Integración con EV3DEV 52
4.8.3 Configuración de la ciudad física 56
4.8.4 Configuración del sistema operativo EV3DEV 57
4.8.5 Componentes utilizados 58
4.8.6 Ensamble general 58
4.8.7 Esquema y dimensiones 61
4.9 Ejemplos aplicables al robot Lego EV3 - DVC-lin3 62
4.9.1 Estructuras de selección simple o condicional 62
4.9.2 Estructuras de iteración 62
4.9.3 Subprogramas. Procedimientos 63
5. Conclusiones 65

5.1 Trabajos futuros 66
6. Herramientas utilizadas..... 67
6.1 Codificación y programación 67
6.2 Documentación y almacenamiento..... 67
6.3 Edición de multimedia 67
6.4 Prototipos 68
6.5 Conectividad..... 68
7. Referencias..... 69
7.1 Bibliografía 70
7.2 Tesis consultadas..... 71
7.3 Páginas web..... 71
Anexos..... 73
I - Código fuente interprete robot..... 73
II - Componentes utilizados 77
III - Ensamble del robot DVC-Lin3 82

Índice de tablas

Tabla 2.1: Tipos de parámetros aceptados 12
Tabla 2.2: Actuales y nuevas funciones primitivas del intérprete DaVinci-C 14
Tabla 2.3: Definición de las nuevas primitivas 17
Tabla 3.1: Clasificación de los robots según T.M. Knasel 20
Tabla 3.2: Comparativa de los robots Lego Mindstorms 33
Tabla 4.1: Palabras reservadas de DaVinci-C 38
Tabla 4.2: Protocolo de comunicación 44
Tabla 4.3: Definición de la implementación del robot 48
Tabla 4.4: Interfaz ev3dev para el sensor de color 54
Tabla 4.5: Interfaz ev3dev para el sensor infrarrojo 56

Índice de figuras

Figura 2.1: Actividad de una función 11
Figura 3.1: Arquitectura de los robots..... 22
Figura 3.2: Evolución de Lego Mindstorms..... 32
Figura 4.1: Esquema general del proyecto 35
Figura 4.2: Arquitectura cliente-servidor general 36
Figura 4.3: E.D.I.S. (Entorno de Desarrollo Integrado Simple) 37
Figura 4.4: Maquetado de la ciudad física..... 37
Figura 4.5: Ejemplo de una posible implementación para la primitiva MOVER..... 40
Figura 4.6: Ejemplo de robot con ruedas direccionales - "Andromina ON ROAD" 41
Figura 4.7: Comunicación de los intérpretes 45
Figura 4.8: Diagrama de clases del intérprete del robot 47
Figura 4.9: Especificación de la ciudad física 49
Figura 4.10: Componentes del kit Lego Mindstorms EV3 52
Figura 4.11: Configuración de la ciudad física a través de la interfaz gráfica..... 56
Figura 4.12: Lego Digital Designer - Ensamble del prototipo..... 59
Figura 4.13: Vista perspectiva DVC Lin3..... 60
Figura 4.14: Vista lateral izquierda DVC Lin3 60
Figura 4.15: Vista superior DVC Lin3 61

Capítulo 1 - Introducción

Desde los inicios de la computadora y los sistemas informáticos, se hizo necesario contar con recurso humano capacitado para su programación. Durante el cursado de la carrera Licenciatura en Informática, en la Facultad de Ingeniería de la Universidad Nacional de la Patagonia San Juan Bosco, he aprendido los conceptos básicos y avanzados de la programación. Estos conceptos despertaron tempranamente mi interés, tanto en la forma de aprenderlos como en la forma en que se enseñaban. Por esa razón acepte mi participación como integrante en el proyecto “Diseño y Aplicación de Estrategias para la Enseñanza Inicial de la Programación” y comenzar a realizar mi tesis con fundamentos que apoyen dentro de este proyecto.

Tanto la tecnología como los lenguajes de programación y la forma de enseñarlos fueron avanzando. Desde el Plankalkül, primer lenguaje de programación de alto nivel, creado a fines de la Segunda Guerra Mundial, hasta los actuales, han sufrido muchos cambios, como así también la forma de enseñarlos, la que ha evolucionado radicalmente. La metodología de aprendizaje y la didáctica empleada para la enseñanza, han tenido una significativa influencia en la formación del estudiante, ya sea en los niveles iniciales o avanzados.

Mediante este trabajo de investigación, se pretende proponer mejoras en la forma de enseñar los conceptos básicos y fundamentales de la programación estructurada, basándose en dos puntos fundamentales: Por un lado, mejorando e incorporando nuevas primitivas y funciones a un lenguaje de programación ya utilizado por el estudiante de los primeros años de la carrera de Sistemas, y por otro lado, despertando el interés de los alumnos, desde un ámbito que está presente en nuestra cotidianeidad: la robótica educativa.

Según la RAE (Real Academia Española), la robótica es una técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales. De las diferentes ramas existentes, en esta tesis se hará uso principalmente a la robótica educativa, que según el autor Ruiz Velasco (Ruiz Velazco E, 2004) la definen como una disciplina que tiene por objeto la concepción, creación y puesta en funcionamiento de prototipos robóticos y programas especializados con fines pedagógicos. En el capítulo 3 se describe el tema en profundidad

Algunas de las ventajas de su incorporación en la enseñanza inicial de la programación pueden resumirse en:

- A través de la robótica educativa se pretende enseñar un sistema de aprendizaje multidisciplinario, que se basa en la resolución de problemas e intenta que el estudiante proponga soluciones aplicando los conocimientos de varias disciplinas. De esta manera se ponen a prueba sus habilidades de gestión, autonomía, liderazgo y comunicación.

- La robótica educativa permite el desarrollo del pensamiento lógico, computacional y matemático para la resolución de problemas, eje fundamental en la programación.

- Programando robots se logra fomentar la creatividad, además de permitir aumentar algunas características fundamentales en nuestra disciplina, como son la curiosidad y el interés por resolver desafíos cada vez más complejos.

1.1 Objetivos

1.1.1 Objetivo general

El objetivo general de esta tesis es generar un producto para motivar al estudiante principiante, vinculado a las carreras de informática, para que alcancen o potencien el pensamiento computacional y crítico para la resolución de problemas a través de la robótica educativa. Mediante el uso de robots físicos se impulsará de una manera motivadora la enseñanza de la programación permitiendo visualizar de manera concreta, en una ciudad física, los algoritmos realizados por los estudiantes.

1.1.2 Objetivos específicos

- Extender el lenguaje de programación DaVinci Concurrente, también llamado DaVinci-C, incorporando nuevas primitivas y funciones.
- Realizar una nueva implementación de la ciudad que refleje la ejecución de algoritmos desarrollados en el lenguaje DaVinci-C a través de robots físicos.
- Programar un robot físico para que realice acciones que serán comunicadas por el intérprete.
- Establecer un protocolo estandarizado de comunicación entre el intérprete y el robot físico.
- Diseñar y, posteriormente, realizar una ciudad física en la que el robot pueda desplazarse.
- Mostrar mediante ejemplos desarrollados en DaVinci-C, la reacción de un robot.

1.2 Alcance de la tesis

Integrar la robótica educativa como un método más de enseñanza en los estados iniciales de la programación, estableciendo los mecanismos necesarios para que el estudiante pueda comprender, de manera análoga sus conceptos básicos. A través de los nuevos procesos de enseñanza y de aprendizaje se espera potenciar la motivación de los alumnos ingresantes a la carrera, al resolver los problemas utilizando la computadora.

Por otra parte su aplicación, en talleres con alumnos del nivel medio de escolaridad, permitirá despertar vocaciones tempranamente, para así mejorar los índices de ingresos de nuevos alumnos a la carrera.

1.3 Lenguajes de programación - Breve síntesis

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser llevados a cabo por máquinas computadoras.(Rodríguez Sala J. J., 2003) Con el tiempo, los lenguajes fueron evolucionando hasta llegar a ser de alto nivel como en estos días. A comienzos del siglo XIX, con la aparición de la máquina del telar de Jacquard, la cual utilizaba tarjetas perforadas para representar movimientos en un brazo, se registró la primera máquina programable de la historia. El tiempo pasó hasta llegar a los lenguajes de programación de alto nivel que se utilizan en la actualidad en la mayoría de los desarrollos informáticos.

Según su historia, los lenguajes de programación se pueden categorizar en tres grupos. El primero, llamado el *lenguaje máquina,* fue creado alrededor de la década del 40 y solo era directamente interpretable por un circuito microprogramable. El segundo, denominado *lenguajes de bajo nivel*, hacía referencia a los que actuaban de manera directa en comunicación con el hardware de la máquina y estaban condicionados por la estructura física de las computadoras de su época. Actualmente muchos sistemas operativos son creados por ellos. Finalmente, el último grupo pertenece a los *lenguajes de alto nivel*, que pueden realizar algoritmos adecuados a las capacidades cognitivas de los seres humanos; otra de sus características es que necesitan un compilador o un intérprete que lo traduzca a un lenguaje de bajo nivel para que la computadora pueda entenderlo.

Es posible hacer otra clasificación de los lenguajes de programación según sus paradigmas. Pueden ser distinguidos como *lenguajes imperativos, declarativos, funcionales, orientados a objetos, orientados al problema, y naturales*. Estos paradigmas son solo proposiciones tecnológicas que fueron aceptadas por los desarrolladores a través del tiempo, los cuales se limitan a resolver problemas definidos. Muchos lenguajes pueden basarse en multiparadigmas, es decir, soportan más de un paradigma de los antes mencionados.

Con los lenguajes de programación se crean los programas informáticos o de computadoras. Se define un programa informático o programa de computadora a una secuencia de instrucciones, escritas en el lenguaje de programación para realizar una tarea específica en una computadora.

Existen diferentes tipos de programas informáticos. Entre ellos se encuentran los programas de *sistemas* o de base, los de *programación* y los de *aplicación.*
Los de *sistemas* son los que llevan el control del funcionamiento de la computadora, ejemplo los sistemas operativos. Los de *programación,* que son los programas utilizados para programar o sea realizar otros programas, por ejemplo DaVinci-C. Los de *aplicación,* son los programas diseñados para o por los usuarios para facilitar la

realización de tareas específicas en la computadora, como pueden ser las aplicaciones ofimáticas

La programación se puede utilizar en infinidad de situaciones. Los programas controlan todo tipo de sistemas, desde robots inteligentes que viajan a otros planetas hasta los electrodomésticos de uso diario que se encuentran en el hogar. Para programar se necesitan conocer algunos conceptos básicos de programación, independientemente del tipo de lenguaje que se utilice. Entre los que se destacan los conceptos de estructuras de control, funciones, procedimientos, variables, modularidad, tipos de datos, lógica y abstracción.

1.4 Enseñanza de la programación

Programar es un proceso que consiste en crear un conjunto de instrucciones en un determinado lenguaje el cual le indica a una computadora cómo debe realizar una tarea específica.

Desde la corporación Microsoft[1], se afirma que los jóvenes pueden beneficiarse de aprender ciencias de la computación básica, independientemente de la profesión que elijan, dado que son parte de la vida cotidiana. Afirmando que hoy en día la programación es la nueva alfabetización.

Microsoft cree que todos los jóvenes deben aprender los principios básicos para programar porque les da las herramientas para ser emprendedores, desarrollar habilidades de pensamiento crítico, lógica y solución a problemas. Han enumerado cuatro beneficios:

1. Desarrolla habilidades para la solución de problemas lógicos. Estimula la creatividad y el pensamiento crítico. Todas estas son habilidades del siglo XXI requeridas en cualquier sector.

2. Mayores oportunidades de empleo. Actualmente, no hay suficiente talento para cubrir la demanda laboral en el sector tecnológico.

3. Proporciona mayor capacidad de adaptación al estilo de vida actual. El mundo depende de la tecnología cada vez más. Con el fin de alcanzar su pleno potencial, los jóvenes necesitan tener una comprensión de cómo funciona la tecnología y cómo hacer que funcione para ellos.

4. El emprendimiento se vuelve una opción tangible. Grandes negocios han surgido gracias al conocimiento del lenguaje de programación. Emprendedores han sabido capitalizar su conocimiento a favor de la creación y crecimiento de sus negocios.

[1] "Microsoft en la Educación." [En línea]. Disponible en: https://www.microsoft.com/es-es/education/. [Último acceso: Junio 2018].

Ya desde el año 1965 se predecía que en el futuro los programadores iban a ser muy necesarios para el crecimiento global, así lo mencionaba la revista popular Science en Estados Unidos en enero de dicho año, la cual hacía referencia a la siguiente cita: "*Se necesitan 500.000 personas que sepan programar. Si Ud. sabe hablarle a una computadora, entonces tiene una excelente oportunidad de trabajo con gran futuro*". (Englebardt S. L., 1965) A pesar de que pasaron muchos años, esta frase no perdido vigencia, la demanda de programadores continúa siendo exponencial a medida que van pasando los años, al mismo paso que la tecnología va evolucionando. Desde los inicios de la computadora, antes definida como un gran aparato que solo tenían las empresas multinacionales, hasta la actualidad, en que pasó a ser un pequeño dispositivo tecnológico que tiene y manipula diariamente el común de las personas, la tendencia no cambió su rumbo.

Actualmente son varias y diversas las formas de aprender programación, el acceso a internet ha abierto un universo de posibilidades para la enseñanza, plataformas web como "codeyear"[2] o "codeacademy"[3] ofrecen la posibilidad de aprender los primeros conceptos de la programación en simples pasos. En Argentina, según la Secretaría de Políticas Universitarias en su sitio web informa que existen 123 universidades entre públicas y privadas en las cuales hay 361 carreras vinculadas a la programación.

1.4.1 Programación en los niveles iniciales

Desde los niveles iniciales, en muchas escuelas públicas y privadas se incentiva la enseñanza de los principios básicos de la programación. Fomentado por el Ministerio de Educación de la Nación y por otras organizaciones, a través de programas tales como Program AR[4], Fundación Telefónica[5], PLANIED (Plan Nacional Integral de Educación Digital)[6], COFECyT (Consejo Federal de Ciencia y Tecnología)[7], entre otros, se brinda al estudiante las herramientas necesarias para desarrollar su pensamiento computacional y despertar la motivación y conciencia necesaria sobre la importancia de la programación.

Según muchos expertos la programación es el idioma del futuro, su aprendizaje influye en muchos aspectos, principalmente en el laboral. El avance de la tecnología lleva a que cada vez existan muchos más sistemas automatizados, cada uno posee una

[2] "Code Year | Codecademy." [En línea]. Disponible en: https://www.codecademy.com/es/tracks/code-year. [Último acceso: Junio 2018].
[3] "Codecademy: Learn to Code - for Free." [En línea]. Disponible en: https://www.codecademy.com/es. [Último acceso: Junio 2018].
[4] "Program.AR." [En línea]. Disponible en: http://program.ar/. [Último acceso: Junio 2017].
[5] "Fundación Telefónica." [En línea]. Disponible en: https://www.fundaciontelefonica.com/. [Último acceso: Junio 2018].
[6] "Plan Nacional Integral de Educación Digital (PLANIED) | SITEAL/TIC." [En línea]. Disponible en: http://www.tic.siteal.iipe.unesco.org/politicas/1778/plan-nacional-integral-de-educacion-digital-planied. [Último acceso: Junio 2018].
[7] "COFECyT - Argentina.gob." [En línea]. Disponible en: https://www.argentina.gob.ar/cofecyt. [Último acceso: Junio 2018].

programación interna cuya existencia muchas veces se ignora porque suele ser mínima.

Por las razones mencionadas, y muchas otras, se hace imprescindible la enseñanza de la programación desde temprana edad para que las futuras generaciones desarrollen el pensamiento computacional y lógico, como así también los conceptos básicos de abstracciones y resolución de problemas.

Capítulo 2 - DaVinci Concurrente

DaVinci Concurrente o DaVinci-C es un lenguaje de programación que, junto a un entorno de desarrollo integrado para la enseñanza de programación estructurada y concurrente (Aguil Mallea D. E., 2012 - Tejero C. G., 2013), fue desarrollado por los docentes, licenciados Daniel Aguil Mallea y Carlos Germán Tejero, como trabajo de sus tesis de grado. Esa investigación tuvo como objetivo modificar la herramienta Visual DaVinci (Champredonde R., 1997), desarrollada por el docente Raúl Champredonde de la Facultad de Informática de la UNLP y que era utilizada por estudiantes de los años iniciales de las carreras de sistemas de la UNPSJB, sede Ushuaia. Los autores, se dispusieron a desarrollar una nueva herramienta con características y funciones adicionales, que se adaptara a los cambios tecnológicos y a las necesidades propias de la materia que la utilizaba.

La nueva herramienta tenía por objetivo fortalecer la base en el aprendizaje del estudiante, tanto en la etapa inicial de la programación estructurada como en la introducción a los conceptos básicos de la programación concurrente. Actualmente es utilizada por los estudiantes de la Universidad Nacional de Tierra del Fuego en las materias iniciales de programación, brindando las herramientas necesarias para adentrarse en el mundo de la programación y la programación concurrente.

2.1 Características del actual desarrollo

La herramienta DaVinci-C fue pensada, especialmente, para que el estudiante ingresante a las carreras de sistemas se inicie en el aprendizaje de la programación, por ende en su desarrollo fue dotado de características que la hacen única y con una aplicación específica.

Sus características principales son las siguientes:

- Se adapta a las necesidades del estudiante sin conocimientos previos en programación.
- Interfaz y lenguaje en castellano.
- Posee implementación multiplataforma.
- Permite configurar e incluir distintos tipos de elementos (flores, papeles y obstáculos) para plantear diversos problemas a resolver en el IDE
- Permite la incorporación de varios robots dentro de la ciudad, trabajando en forma concurrente.
- Permite visualizar la ejecución de los programas ayudando al estudiante a comprender el resultado del algoritmo desarrollado.
- Proyecto libre. Todo el código fuente y su documentación se pueden descargar. También permite que se contribuya para su mejora a través del sitio web colaborativo para proyectos de software llamado sourceforge.

2.2 Ventajas y desventajas

Las características citadas en el punto 2.1 forman parte de las ventajas del actual desarrollo del DaVinci-C.

Las desventajas pueden resumirse en:

- Una visualización, más real, de las actividades de los robots, sobre una ciudad física resultaría mucho más motivadora para los alumnos, al menos en el aprendizaje inicial de las estructuras de control. (Las características de la nueva aplicación incorporada, será desarrollada a partir del capítulo III)

- Por otra parte, si bien DaVinci-C cumple con los requerimientos para el cual fue desarrollado, las “funciones”, como un tipo de subprogramas, que son un punto importante para la enseñanza de la programación modular, no están disponibles. Por este motivo se hace necesaria su implementación. Este tema y la justificación de otras mejoras agregadas al lenguaje se profundizará en los capítulos siguientes.

2.3 Características del lenguaje e intérprete DaVinci-C

Las características del actual intérprete se presentan a continuación:

- El lenguaje de programación posee una estructura simple y clara.
- Posee todas las estructuras de control de flujo, entre ellas “repetir”, “mientras”, “si” y “sino”.
- Tiene implementado distintos planificadores para el manejo de concurrencia, entre ellos los más comunes FIFO, Aleatorio, Secuencial y Round Robin.
- Posibilidad de la lectura de variables en tiempo de ejecución.
- Cuenta con estructuras, tipos de datos y primitivas para el manejo de concurrencia.
- No es sensible a las mayúsculas ni a las minúsculas.
- No se limita al sangrado como mecanismo sintáctico para delimitar los bloques de código, sino que lo hace con las palabras clave “comenzar” y “fin”.
- Contiene mensajes de errores sintácticos bien definidos.

2.4 Mejoras propuestas al lenguaje

Luego de un análisis realizado sobre el uso del lenguaje DaVinci-C y en base a las experiencias realizadas con los alumnos y los docentes de la asignatura “Expresión de Problemas y Algoritmos”, se observó la necesidad de realizar algunas mejoras al intérprete dotándolo de nuevas herramientas.

En este trabajo se incorporaron específicamente el desarrollo para la implementación de las nuevas primitivas:

- El subprograma **tipo función**, que podrá ser definido por el usuario según sus necesidades,
- Un nuevo grupo de funciones definidas por el lenguaje (todas de tipo función).

2.4.1 Justificación de las mejoras propuestas: importancia del uso de las funciones

El concepto de modularización (subprogramas) es un aspecto fundamental para el desarrollo de algoritmos. Si bien sus ventajas pueden observarse con más claridad en la resolución de algoritmos complejos, la enseñanza de los conceptos y sus aplicaciones debe hacerse tempranamente.

Los mismos se ven como instrumento de modularización, se los declara una sola vez, pero pueden ser invocados varias veces desde otras partes del programa. Son una parte fundamental en la programación estructurada, ya que permiten la descomposición de un problema en subproblemas más pequeños y controlables de manera independiente.

Entre las ventajas del uso de la modularización, puede mencionarse el concepto de reuso, que reduce el tiempo de codificación y ayuda en la depuración, disminuyendo así el tiempo total de desarrollo; otro aporte fundamental es la posibilidad de resolver problemas de menor complejidad, que en general son más sencillos y tienen una tarea específica e independiente dentro del principal problema a resolver. Permitiendo así, una mejor comprensión de la resolución del problema. Recordemos que si la resolución del problema es fácil de entender, entonces será más fácil encontrar errores o hacer modificaciones.

Los subprogramas son secciones de código separado o no, del programa principal que tienen un comportamiento independiente facilitando su diseño, seguimiento y corrección de errores. (Berlanga Llavori R. y Quereda J. M. I., 2000) El concepto de subprograma es producto de la evolución del antiguo concepto de subrutina, presente en lenguajes como Ensamblador, Basic o las primeras versiones de Fortran.
Un factor importante a tener en cuenta en la definición de subprogramas es que, en general, pueden ser de dos tipos: Procedimientos o Funciones. Por otra parte, para cada tipo, pueden ser primitivas dentro del lenguaje, o declaradas por el programador, atendiendo a sus necesidades.

La diferencia entre los tipos, funciones y procedimientos, radica en que una función se usa para devolver un único valor de un tipo de dato simple, a través de su nombre. Los procedimientos pueden devolver varios o ningún valor, utilizando los parámetros. Ambos pueden ser invocados desde cualquier parte programa principal, hilo u otro subprograma, haciendo referencia a través de su identificador.

Resumiendo puede enunciarse que la utilidad específica del tipo de subprograma, es que permite entender con más claridad la lógica utilizada para resolver el problema,

potenciando la facilidad de depuración y búsqueda de errores en un programa. El mantenimiento y la modificación de la programación se facilitan.
Por lo expuesto, es fundamental la enseñanza de estos conceptos en los cursos iniciales de programación.

El lenguaje de programación DaVinci-C soportaba únicamente la utilización de procedimientos en un área especial identificada por la palabra reservada "subprogramas". Si bien contemplaba funciones primitivas definidas en el lenguaje, no ocurría lo mismo con la definición de funciones realizadas por el programador.

2.4.1.1 Parámetros de los subprogramas

Los subprogramas, procedimientos y/o funciones, establecen su comunicación con el programa principal y/o cualquier otro subprograma que los invoque, a través del pasaje parámetros, que como veremos pueden ser de distinto tipo.

Los parámetros y argumentos, son la forma de comunicación entre el programa principal y los subprogramas, a través de su invocación. En el lenguaje DaVinci-C podemos distinguir a tres tipos parámetros como forma de comunicación en los subprogramas.

- **Parámetros de entrada**: permiten ingresar información en el subprograma como una copia del mismo, si sufriera cambios no se verían reflejados en el ámbito de su invocación, ya sea el programa principal, un hilo u otro subprograma.
- **Parámetros de salida**: permiten comunicar la información generada dentro del subprograma, por ende, se utiliza el parámetro para devolver un resultado. El valor inicial del parámetro no influye en el resultado final.
- **Parámetros de entrada/salida**: permiten ingresar información hacia el subprograma y retornar las modificaciones dentro de la misma referencia con cual fue invocada.

2.5 Implementación del subprograma tipo función en el DaVinci-C

Una función, definida por el programador es un segmento de código, que realiza una tarea específica y devuelve, al ser invocada, un único valor de tipo simple, asociada a su nombre.
Como cualquier subprograma, puede recibir parámetros tanto de entrada, salida o de entrada/salid. Debe destacarse que estos dos últimos son de **uso excepcional**, dado que el uso de este tipo de subprograma, debe hacerse cuando se necesita retornar un valor en función a los parámetros que recibe, los que deberían **ser solo de entrada.**

Figura 2.1: Actividad de una función

El uso de funciones otorgaría mayor claridad, sencillez y legibilidad cuando el estudiante desarrolle sus algoritmos y plantee sus soluciones a los problemas presentados. Otro factor importante es que los acercará mínimamente a los lenguajes funcionales cuyo paradigma se basa en programación declarativa, que es la tendencia actual en uso.

La extensión desarrollada respeta toda la especificación del actual lenguaje, entre ellos el ámbito, la gramática, la estructura léxica, los tipos de datos, valores y operaciones del intérprete DaVinci-C.

En pseudolenguaje una función tendría la siguiente forma:

```
FUNCION <nombre> (parámetro 1: tipo1, ..., parámetro n: tino n):
<tipo-retorno>
VARIABLES
      <declaración de variables locales a la función>
COMIENZO
      <instrucciones / sentencias>
RETORNAR <expresión>
FIN
```

Donde

<nombre>: es el identificador de la función,
parámetro i: representa el parámetro i-ésimo de la función,
tipo i: representa el tipo i-ésimo parámetro de la función,
<tipo-retorno>: representa el tipo de dato que devolverá la función,
<instrucciones>: bloque de instrucciones/sentencias de la función,
<expresión>: representa el valor que devuelve la función.

Dentro de todo el código fuente del programa desarrollado, las funciones pueden ser llamadas desde:

- sentencias del programa principal,
- sentencias propias de la función (funciones recursivas),
- dentro de cualquier otro subprograma (función o procedimiento) que la invoque

La estructura del lenguaje de programación DaVinci-C es la siguiente:

```
programa <nombre>

variables
      <declaraciones globales>

subprogramas
      <subprogramas>
```

```
variables
      <declaraciones locales>

comenzar
      <instrucciones>
fin
```

Donde

<*nombre*>: nombre del programa principal,

<*declaraciones globales*>: declaración de variables globales del programa principal,

<*subprogramas*>: declaración de subprogramas, tanto procedimientos como funciones,

<*declaraciones locales*>: declaración de variables locales del programa principal,

<*instrucciones*>: bloque de instrucciones del programa principal.

2.5.1 Especificación en el lenguaje del tipo función

La extensión que se realizó al lenguaje permite declarar y utilizar funciones como parte de los subprogramas. Para definirlas se utilizará la palabra reservada “funcion” seguida del nombre que la identificará, con sus correspondientes parámetros en el caso de ser necesarios.

Como se mencionó anteriormente, el lenguaje soporta tres modos de comunicación para los parámetros que pueden ser utilizados por las funciones definidas por el estudiante; si bien suelen utilizar solo parámetros de entrada, se decidió dejar el resto para contemplar los **casos excepcionales**. Los parámetros mencionados se definen de la siguiente manera:

Nombre	Palabra reservada	Descripción
Parámetros de entrada	en	El resultado depende del valor del parámetro, pero este no modifica el valor original del programa principal.
Parámetros de salida	sa	El resultado no depende del valor del parámetro, solo lo utiliza para devolver un resultado que se procesa en la función.
Parámetros entrada/salida	es	El resultado depende del valor parámetro y también se utiliza para devolver un resultado que se procesa en la función.

Tabla 2.1: Tipos de parámetros aceptados

Respetando la estructura de los lenguajes de programación convencionales, la sintaxis de la declaración de las funciones se definió de la siguiente manera:

```
funcion <nombre> (parámetro 1: tipo1, ..., parámetro n: tipo n):
<tipo-retorno>
      variables
            <declaración de variables locales a la función>
      comenzar
            <instrucciones>
            retornar <expresión>
      fin
```

Donde

<nombre> : es el nombre de la función,
parámetro i : representa el parámetro i-ésimo de la función,
tipo i : representa el tipo i-ésimo parámetro de la función,
<tipo-retorno> : representa el tipo de dato que va a devolver la función,
<instrucciones> : bloque de instrucciones de la función,
<expresión> : representa el valor que devuelve la función.

Para ejemplificar se muestra la utilización de un subprograma con el uso de funciones en el lenguaje DaVinci-C.

```
programa ejemploFuncion

subprogramas
  funcion volumen(en l1:numero; en l2:numero; h:numero) : numero
    variables
      v:numero
    comenzar
      v := l1 * l2 * h
      retornar v
    fin

variables
   resultado:numero

comenzar
  resultado:= volumen(2,5,3)
  informar(‘volumen: ’, resultado)
fin
```

En el ejemplo anterior, la ejecución del programa comienza con la asignación de una variable local denominada “resultado”, a esta se le asigna el resultado de una función llamada “volumen”, la cual recibe tres parámetros de entrada y realiza un cálculo con ellos. Luego de ese cálculo, la función invocada retorna un valor producto de un cálculo interno. El programa principal vuelve a tomar el control e informar el valor de la variable “resultado”.

2.6 Funciones definidas por el lenguaje que se incorporan

A continuación se identifican las funciones primitivas que posee el actual lenguaje DaVinci-C, así como también, las nuevas que se incorporan para la manipulación de cadenas, números y valores lógicos.

Actuales funciones primitivas	aleatorio, longitud, sustraer, numeroATexto, textoANumero, logicoATexto, hayRobot
Nuevas funciones primitivas	reemplazar, comienzaCon, terminaCon, minuscula, mayuscula, cortar, indiceDe, letraAleatoria, raiz, potencia, mayorValor, seno, coseno, tangente, logaritmo

Tabla 2.2: Actuales y nuevas funciones primitivas del intérprete DaVinci-C

2.6.1 Definición de las nuevas funciones primitivas del lenguaje

Se incorporaron nuevas primitivas al lenguaje DaVinci-C, focalizándose principalmente el uso de manejo de caracteres y funciones matemáticas.

Las mismas fueron seleccionadas junto al equipo de cátedra, que las propuso en relación al abanico de ejercicios que podrían incluirse en las guías de trabajos prácticos a resolver por los alumnos.

A continuación se describe cada una de las funciones incorporadas, con su sintaxis correspondiente.

Primitiva	Descripción
reemplazar	A partir de una cadena de texto, reemplaza todas las cadenas de texto que coinciden en el primer parámetro por la cadena de texto ingresada en el segundo parámetro. **Sintaxis**: reemplazar(en cadena:texto, en v1:texto, en v2:texto) retorna texto **Ejemplo**: ● original := “esto es una espada” ● prueba := reemplazar(original, “es”, “xo”) ● informar(prueba) *salida → xoto xo una xopada*
comienzaCon	Devuelve un valor lógico comprobando si la cadena de texto

	ingresada empieza o no con el segundo parámetro ingresado. **Sintaxis**: comienzacon(en cadena:texto, en v1:texto) retorna logico **Ejemplos**: ● log := comienzacon(original, “abc”) ● si (comienzacon(original, “abc”)) infomar(“el texto ingresado comienza con abc”)
terminaCon	Devuelve un valor lógico comprobando si la cadena de texto ingresada termina o no con el segundo parámetro ingresado. **Sintaxis**: comienzacon(en cadena:texto, en v1:texto) retorna logico **Ejemplos**: ● log := comienzacon(original, “abc”) ● si (comienzacon(original, “abc”)) infomar(“el texto ingresado comienza con abc”)
minuscula	Devuelve la cadena de texto ingresado en minúscula en función a la cadena ingresada por el parámetro de entrada. **Sintaxis**: minuscula(en cadena:texto) retorna texto **Ejemplos**: ● cadena := minuscula(original) ● informar(minuscula(original))
mayuscula	Devuelve la cadena de texto ingresado en mayúscula en función a la cadena ingresada por el parámetro de entrada. **Sintaxis**: mayuscula(en cadena:texto) retorna texto **Ejemplos**: ● cadena := mayuscula(original) ● informar(mayuscula(original))
cortar	Devuelve la cadena de texto ingresado sin los espacios del principio y fin que pueda llegar a tener en función a la cadena ingresada por el parámetro de entrada. **Sintaxis**: cortar(en cadena:texto) retorna texto **Ejemplos**: ● cadena := cortar(“ prueba ”) ● informar(cortar(“ prueba ”))
indiceDe	Devuelve un valor entero en la primera posición que se encuentra el

	carácter o cadena de caracteres en función a un texto ingresado como primer parámetro de entrada. **Sintaxis**: indicede(en cadena:texto, en t1:texto) retorna numero **Ejemplos**: ● valor := indicede(cadena,“a”) ● informar(indicede(cadena,“a”))
letraAleatoria	Devuelve una letra aleatoria del alfabeto. **Sintaxis**: letraaleatoria() retorna letra **Ejemplos**: ● caracter := letraaleatoria() ● informar(letraaleatoria())
raiz	Devuelve la raíz cuadrada (entera) del valor en función al valor del parámetro de entrada. **Sintaxis**: raiz(en valor:numero) retorna numero **Ejemplos**: ● n := raiz(9) ● informar(raiz(9))
potencia	Devuelve como resultado la potencia en función al valor y el exponente que se ingrese a través de los parámetros de entrada. **Sintaxis**: potencia(en valor:numero, en exponente:numero) retorna numero **Ejemplos**: ● n := potencia(3,2) ● informar(potencia(3,2))
mayorValor	Devuelve el mayor valor de dos números enteros ingresados en los parámetros de entrada. **Sintaxis**: mayorvalor(en valor1:numero, en valor2:numero) retorna numero **Ejemplos**: ● n := mayorvalor(30,25) ● informar(mayorvalor(30,25))
seno	Devuelve la función trigonométrica seno (entera) en función del valor del parámetro de entrada. **Sintaxis**:

	seno(en valor:numero) retorna numero **Ejemplos**: ● n := seno(45) ● informar(seno(45))
coseno	Devuelve la función trigonométrica coseno (entera) en función del valor del parámetro de entrada. **Sintaxis**: coseno(en valor:numero) retorna numero **Ejemplos**: ● n := coseno(45) ● informar(coseno(45))
tangente	Devuelve la función trigonométrica tangente (entera) en función del valor del parámetro de entrada. **Sintaxis**: tangente(en valor:numero) retorna numero **Ejemplos**: ● n := tangente(45) ● informar(tangente(45))
logaritmo	Devuelve el logaritmo (entera) en función del valor del parámetro de entrada. **Sintaxis**: logaritmo(en valor:numero) retorna numero **Ejemplos**: ● n := logaritmo(185) ● informar(logaritmo(185))

Tabla 2.3: Definición de las nuevas primitivas

Capítulo 3 - Robótica educativa

La robótica es una rama de la tecnología que se dedica al diseño, fabricación y aplicación de dispositivos electromecánicos llamados robots, abarcando varias disciplinas como álgebra, mecánica, electrónica, informática, inteligencia artificial e ingeniería de control. (Baturone A. O., 2005) Los robots son dispositivos mecánicos que fueron desarrollados con el principal objetivo de realizar varias tareas previamente programadas para que cubran el trabajo realizado por el hombre.

La robótica educativa, o robótica pedagógica, es el conjunto de actividades pedagógicas que apoyan y fortalecen áreas específicas del conocimiento, desarrollando competencias en el estudiante a través del diseño, desarrollo, construcción y puesta en funcionamiento de robots. (Cuero Sandoval E., 2015) Sus objetivos son promover el aprendizaje colaborativo y experimental, desarrollar el conocimiento, la capacidad creativa, de análisis y de síntesis. A través del pensamiento crítico y habilidades manuales se pueden esquematizar modelos para diversas disciplinas curriculares verificando los resultados del proyecto al instante.

Básicamente, cuando se construye un proyecto se produce conocimiento y, aprender a programar, ayuda a organizar el pensamiento computacional. La robótica educativa une ambos objetivos al tiempo que propone la resolución de problemas buscando soluciones creativas, así como también, fomenta el trabajo en equipo e invita a experimentar y obtener resultados que permitan mejorar las propuestas iniciales. No solo se trata de construir y armar robots sino también de que se utilice este recurso tecnológico como factor de motivación para ayudar al estudiante a construir su propio conocimiento.

En Argentina a través de programas nacionales se fomenta la robótica en la educación, tal es el caso del proyecto “Programa Robótica y Tecnología para Educar” pensado para que el estudiante de nivel inicial tenga una noción de las nuevas tecnologías y pueda estimular un aprendizaje innovador y colaborativo.

3.1 Historia

La robótica existe desde hace miles de años; antes los robots eran conocidos como autómatas. Desde el inicio, los seres humanos han querido crear seres artificiales que se encarguen de realizar tareas repetitivas, pesadas o difíciles. Se puede decir que los primeros autómatas fueron las máquinas que ayudaban al hombre a facilitar su trabajo a través de la fuerza bruta.

A lo largo del tiempo esta rama de la tecnología ha ido creciendo en lo que conocemos actualmente como robótica. La palabra robótica fue usada por primera vez a principios del siglo XX en una obra de teatro, su origen proviene de la palabra checa “robota” que se refiere a servidumbre o trabajo forzado. En los años 50 se patentó el primer robot

industrial perteneciente a George C. Devol[8]. Un momento icónico en la historia de la robótica fue con la aparición de la primera máquina programable de Jacquard, en 1801, la cual marcó el inicio de la programación como vimos en el Capítulo I. A partir del descubrimiento de los semiconductores y el crecimiento del uso de las computadoras empezó un desarrollo más profesional.

Durante la primera y la segunda Revolución industrial, la creación de diversos elementos cambiaron el rumbo de la historia tanto en lo social como en lo político y económico. La potenciación de las máquinas ayudó a optimizar el trabajo antes realizado por el ser humano. La tercera Revolución industrial también ha sido denominada Revolución de la inteligencia por la importancia que la tecnología y las comunicaciones han tenido en la transformación de la industria, entre ellos la creación de las computadoras e internet.

En la era de la información, conocida como era digital o de la informática, a la que se suman las tecnologías de la información y la comunicación, aparecen los robots actuales y la inteligencia artificial que nos muestran nuevas formas de afrontar el mundo laboral, ayudando y facilitando el trabajo humano. Actualmente estamos en la plenitud de la cuarta Revolución industrial, también llamada Industria 4.0, la cual genera cierto temor en las personas al pensar que se las sustituye por máquinas totalmente automatizadas. La tecnología avanza día a día, trayendo como consecuencia una sofisticación cada vez mayor en la construcción de robots.

Hoy la robótica puede encontrarse en la construcción de maquinarias y automóviles, hospitales, empresas de logística, fábricas, aeropuertos, y muchos lugares más. Las aplicaciones de la robótica en sus diferentes áreas son incontables y van creciendo exponencialmente.

3.2 Clasificación de los robots

Más allá de la tipología y clasificación que tengan los actuales robots, la única característica que comparten la mayoría es que son programables, por ende, se necesita una persona física que realice ese trabajo.

En 1986 un investigador de la Universidad de Harvard, T. Michael Knasel (Knasel M 1986), propuso una clasificación de robots en cinco grupos de generaciones, la que a pesar de los años no ha perdido validez y permite tener una visión del pasado, presente y futuro de la robótica. Los grupos de generaciones propuestos fueron los siguientes:

[8] "George Devol - Wikipedia, la enciclopedia libre." [En línea]. Disponible en: https://es.wikipedia.org/wiki/George_Devol [Último acceso: Junio 2018].

Clasificación de los robots según T.M. Knasel				
Generación	**Nombre**	**Tipo de control**	**Grado de movilidad**	**Usos más frecuentes**
1 (1982)	Pick & place.	Fines de carrera, aprendizaje.	Ninguno.	Manipulación, servicio de máquinas.
2 (1984)	Servo.	Servocontrol, trayectoria continua, progresión condicional.	Desplazamiento por vía.	Soldadura, pintura.
3 (1989)	Ensamblado.	Servos de precisión, visión, tacto.	Guiado por vía	Ensamblado, desbarbado.
4 (2000)	Móvil.	Sensores inteligentes.	Patas, ruedas	Construcción, mantenimiento.
5 (2010)	Especiales.	Controlados con técnicas de inteligencia artificial.	Andante, saltarín.	Militar, espacial.

Tabla 3.1: Clasificación de los robots según T.M. Knasel

Una de las clasificaciones más comunes, dada en Wikipedia[9], se basa en la cronología, tiene en cuenta la historia de la robótica y permite observar el paso del tiempo y su evolución.

- **1ª Generación: Robots manipulables**: creados en los años 50, eran multifuncionales con un sistema sencillo que repite tareas programadas y actúan en consecuencia a los sucesos.
- **2ª Generación: Robots de aprendizaje**: desarrollados en los 80, son conscientes de su entorno, disponen de sistemas de control de lazo cerrado.
- **3ª Generación: Robot con control sensorizado**: surgen entre los 80 y 90, cuentan con computadoras que procesan la información y los datos obtenidos a través de sensores.
- **4ª Generación: Robots inteligentes**: son creados entre los años 90 y comienzos del 2000, poseen sensores mucho más sofisticados que mandan información a la computadora y la analiza mediante estrategias complejas de control.

[9] "Robótica - Wikipedia, la enciclopedia libre." [En línea]. Disponible en: https://es.wikipedia.org/wiki/Rob%C3%B3tica. [Último acceso: Junio 2018].

- **5ª Generación: Inteligencia artificial**: desde los inicios del siglo XXI hasta la actualidad se vienen desarrollando robots con esta nueva tecnología que incorpora inteligencia artificial. Utilizan modelos de conducta y una arquitectura de subsunción donde las conductas complejas son el resultado evolutivo de funciones conductuales simples.

3.2.1 Autonomía

La autonomía de los robots, y de cualquier dispositivo electrónico, es la capacidad de autoabastecimiento en un tiempo determinado conectado a una fuente de alimentación propia. Otra definición se basa en observar su capacidad para adaptarse a los cambios y situaciones eventuales que pudiesen ocurrir en su normal funcionamiento.

Entonces, según su autonomía se pueden distinguir a tres tipos de robots:
- teleoperados,
- semiautomáticos,
- automáticos.

3.2.2 Tipos

Se podrían realizar numerosas clasificaciones de robots de acuerdo a sus características y en base a su evolución. En la actualidad existen los siguientes tipos:
- Robots industriales de manipulación
    - automoción,
    - siderurgia.
- Robots de servicios
    - robots domésticos,
    - robots médicos,
    - robots militares,
    - nano robots,
    - juguetes robóticos,
    - robots imprimibles,
    - robots educacionales.

3.3 Arquitectura de los robots

Según la estructura de un sistema de control en un robot (Barber Castaño R. I., 2000) se pueden distinguir los componentes que a continuación se mencionan:
- **Sensores**: se encargan de brindar los datos de entrada y/o de retroalimentación del sistema, estos pueden ser activos o pasivos. Se subclasifican así:
    - sensores de luz,
    - sensores de presión y fuerza,
    - sensores de sonido,

  - sensores para medición de distancia,
  - sensores de posición,
  - sensores de temperatura,
  - sensores de humedad,
  - sensores de velocidad,
  - sensores de magnetismo,
  - sensores de ubicación geográfica,
  - sensores de proximidad.
- **Unidad de control (CPU)**: procesa la información que proporcionan los sensores y toma decisiones de acuerdo con dicha información.
- **Actuadores**: elementos que ejecutan las órdenes de la unidad de control según sus grados de libertad que son los posibles movimientos básicos. Pueden ser eléctricos, mecánicos, hidráulicos o neumáticos.

Figura 3.1: Arquitectura de los robots

La CPU del robot recibe información desde los sensores, la procesa, y ejecuta las órdenes a través de los actuadores según los grados de libertad que posean.

3.4 Aplicaciones de la robótica

Las aplicaciones que se le puede dar a la robótica son infinitas. Cada robot tiene funciones y características particulares. Algunas de sus ramas son las siguientes:

- industria,
- construcción,
- entornos peligrosos,
- medicina y salud,
- minería,
- entornos submarinos,
- vigilancia y seguridad,
- educación.

Tal como se adelantó en el capítulo I, parte de esta tesis se basará en la robótica educativa, y se desarrolla dentro de un proyecto de investigación para la creación de un espacio didáctico tecnológico, que involucra a docentes y estudiantes de las dos carreras de sistemas.

3.4.1 La robótica y su aplicación en la educación

La robótica educativa, o robótica pedagógica, es una disciplina que tiene por objeto la concepción, creación y puesta en funcionamiento de prototipos robóticos y programas especializados con fines pedagógicos. (Ruiz Velasco E. S., 2004)

Uno de los objetivos principales de la robótica educativa es promover el desarrollo del pensamiento lógico formal de los estudiantes en los niveles iniciales y medios. Con su desarrollo se pretende despertar el interés y la motivación en estudiantes de temprana edad a fin de potenciar el pensamiento computacional, conveniente, para abordar estudios superiores de cualquier disciplina y despertar tempranamente posibles vocaciones en algunas de las carreras vinculadas la informática. Estas carreras han sido declaradas entre las carreras estratégicas por el Ministerio de Educación[10] . La demanda de profesionales crece año a año tanto en Argentina como a nivel mundial.
A través de la robótica educativa se desarrolla la capacidad analítica y de resolución de problemas. Uno de los primeros países en promover esta disciplina en los niveles iniciales fue Corea, a través de talleres itinerantes fuera del aula, que dieron buenos resultados. Esto llevó a que en 1998 se modificara su esquema educativo incorporando nuevas actividades que involucraron a la robótica.

La importancia de su aplicación en el proceso educativo de niños y adolescentes es que mejora sus capacidades en distintas áreas y estimula el desarrollo del pensamiento reflexivo, sentido de anticipación, actitudes creativas y la habilidad para la resolución de problemas.

Según un estudio realizado por una importante consultora especializada en el mercado laboral, Snoop Consulting[11], más de la mitad de los niños y niñas que actualmente están estudiando en los niveles iniciales, tendrán, en el futuro, un empleo relacionado con el sector tecnológico. Esto representa un nuevo desafío a nivel educacional y una nueva forma de plantearse el escenario educativo. Evidenciando, de esta manera, la necesidad de incluir nuevos métodos didácticos y educativos para preparar a las nuevas generaciones a afrontar los cambios que se vienen tanto en lo educacional como en lo laboral.

Esta tesis solo abarca un sector específico de la robótica educativa, la cual se encarga del uso de tecnologías de programación y robótica, para la enseñanza de programación a través de robots y sus aplicaciones. De esta manera se intentará motivar al estudiante a programar y razonar de manera lógica para la resolución de problemas.

Las actividades que se basan en robots incrementan el compromiso por el aprendizaje colaborativo y motivan la realización de proyectos determinados. El pensamiento

[10] "Ministerio de Educación | Argentina.gob.ar." [En línea]. Disponible en: https://becasprogresar.educacion.gob.ar/. [Último acceso: Junio 2018].
[11] "Snoop Consulting." [En línea]. Disponible en: https://www.snoopconsulting.com/. [Último acceso: Junio 2018].

computacional que desarrolla el estudiante lo fortalece y le da las herramientas necesarias para afrontar todo tipo de situaciones en la vida cotidiana.

La robótica es una herramienta de aprendizaje. Su combinación con la programación potencia la creatividad, la lógica visual y la capacidad de trabajo en equipo. Es una forma bastante intuitiva de familiarizarse con la electrónica y la programación.

3.5 Placas programables

Las placas programables son dispositivos que contienen entradas y salidas. Estas trabajan conjuntamente con un microcontrolador y/o microprocesador para la realización de proyectos automatizados. Entre ellas se encuentran las placas Arduino, Lego MindStorms, RaspBerry Pi, Matrix ARM, BeagleBoard, Odroid, Banana Pi, Joule 570X, Placa DECA, Thunderboard React, NRF52-DK, Synergy S7 y muchas más.

Del conjunto de todas esas placas se pueden distinguir algunas cuyo propósito principal de creación es la enseñanza. Si bien existe una gran variedad, se optó por elegir las placas que poseían una mayor estandarización y documentación extensiva en la web, las mismas se mencionan a continuación:

- RaspBerry Pi[12]
- Arduino[13]
- Lego Mindstorms[14]

A diferencia de las placas Arduino, las placas RaspBerry Pi poseen un microprocesador que tiene un microcontrolador, asemejándose más a una computadora. Si bien pueden comportarse de igual manera para determinados problemas, cada una fue diseñada para propósitos diferentes. Son conocidas como SBC (Simple Board Computer) y, a diferencia de una computadora normal, llevan todos sus componentes en una sola placa. Otra diferencia importante es que estas llevan un procesador gráfico.

En las placas RaspBerry Pi se requiere un sistema operativo para ejecutar sus instrucciones, en Arduino no es necesario. Se recomienda utilizar este tipo de placas para proyectos con gran procesamiento de datos, que soliciten una interfaz visual con gran resolución, desarrollos con una rápida interfaz de red y multitarea. Si bien estas placas ofrecen grandes particularidades en comparación al resto de similares características, a un costo no muy superior, su desventaja en relación al resto de las placas programables es la complejidad y poca flexibilidad que ofrecen para trabajar con proyectos de hardware.

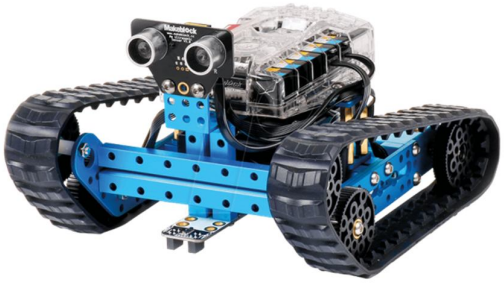
[12] "Raspberry Pi - Teach, Learn, and Make with Raspberry Pi." [En línea]. Disponible en: https://www.raspberrypi.org/. [Último acceso: Junio 2018].
[13] "Arduino." [En línea]. Disponible en: https://www.arduino.cc/. [Último acceso: Junio 2018].
[14] "Home - Mindstorms LEGO.com." [En línea]. Disponible en: https://www.lego.com/en-us/mindstorms. [Último acceso: Junio 2018].

Las características de las placas Arduino y Lego MindStorms se mencionan en los capítulos posteriores.

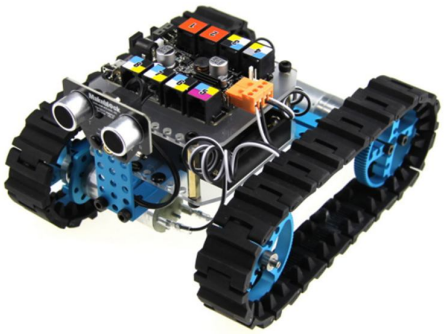
3.5.1 Robots educativos programables

En el mercado global existen diversos robots educativos que son programables. A continuación se presenta un cuadro comparativo con sus características y capacidades:

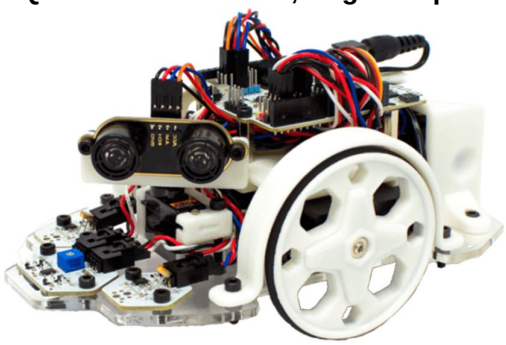
Makeblock mBot V1.1, origen China	**Componentes:** sensor de luz, receptor IR, sensor ultrasónico, sensor de seguimiento de línea, parlante, RGB LED, transmisor IR, dos motores.
	Conectividad: bluetooth (bluetooth Versión) 2.4 GHz wireless serial (2.4 G Versión).
	Placa base: mCore (basado en Arduino Uno).
	Precio: U$D 95.
	Software de programación: mBlock (basada en Scratch 2.0) Arduino IDE.

Makeblock mBot Ranger, origen China 	**Componentes:** dos sensores de luz, sensor de sonido, giroscopio, sensor de temperatura, sensor ultrasónico, seguimiento de línea, parlante.
	Conectividad: Bluetooth - 2.4 GHz wireless serial
	Placa base: Arduino Mega 2560, 256KB flash memory, 8KB SRAM, 4KB EEPROM.
	Precio: U$D 190.
	Software de programación: mBlock (basada en Scratch 2.0) Arduino IDE.

Makeblock Starter Robot Kit , origen China	**Componentes:** 2 x 186 RPM DC Motor

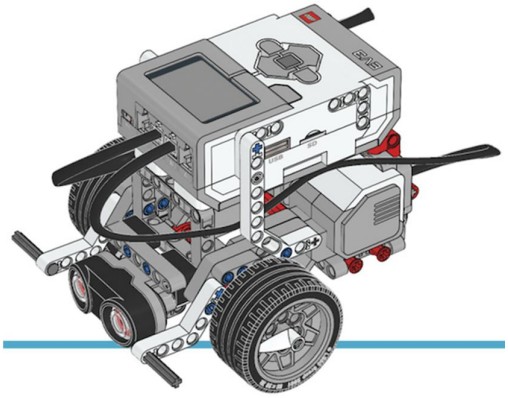
	**Conectividad:** Bluetooth - IR.
	Placa base: Arduino ATmega328P,32 KB flash memory, 2 KB SRAM, 1 KB EEPROM.
	Precio: U$D 150.
	Software de programación: PC-mBlock.

BQ Zum Kit, origen España 	**Componentes:** sensor IR, pulsador, sensor de luz, zumbador, sensor ultrasónicos, potenciómetro, led, miniservo, servo de rotación contínua.
	Conectividad: Bluetooh integrado.
	Placa base: BQ Zum Core Microcontrolador AT mega 328 P.
	Precio: U$D 134.
	Software de programación: APP Bitbloq.

BQ PrintBot Evolution, origen España 	**Componentes:** Zum Core, dos ZUMbloq sensor de luz, ZUMbloq zumbador, ZUMbloq infrarrojo doble, sensor ultrasónico, miniservo, dos servos de rotación contínua.
	Conectividad: puerto micro USB tipo B, módulo bluetooth integrado en la placa controladora.
	Placa base: placa controladora BQ.
	Precio: U$D 123.
	Software de programación: APP Bitbloq.

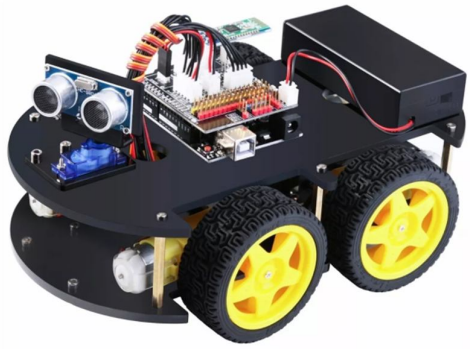
Lego WeDo 2.0	**Componentes:** sensor de inclinación, sensor de movimiento, motor

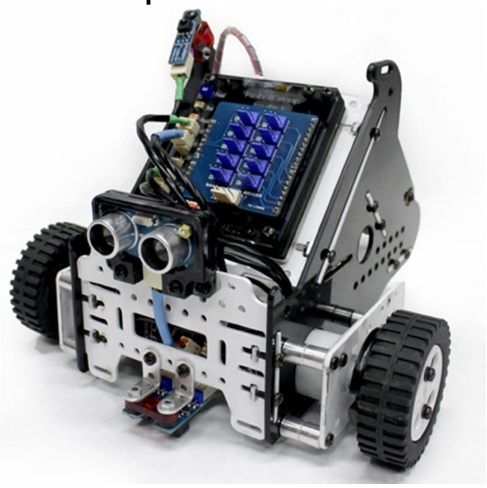
	mediano.
	Conectividad: módulo bluetooth integrado.
	Placa base: Hub inteligente.
	Precio: U$D 190.
	Software de programación: WeDo 2.0 software

Lego Mindstorms EV3 	**Componentes:** sensor de color, sensor táctil, IR sensor, motor grande (x2), motor mediano.
	Conectividad: bluetooth, Wi-Fi a través del puerto USB del módulo central.
	Placa base: bloque programable EV3
	Precio: U$D 490.
	Software de programación: EV3 programmer, proyecto ev3dev.

SunFounder Raspberry Pi Smart Robot Car Kit 	**Componentes:** sensor ultrasónico, sensor de seguimiento de línea, sensor de luz seguidor ultrasónico.
	Conectividad: WI-FI.
	Placa base: robots HATS.
	Compatibilidad: Raspberry Pi model B+, dos model B and tres model B.
	Precio: U$D 120.
	Software de programación: Python SMBUS.

Qbot Programmable Smart Robot Car	**Componentes:** sensor ultrasónico, dos baterías de litio, pantalla y kit de apoyo.
	Conectividad: control remoto infrarrojo.
	Placa base: motor driven board.
	Precio: U$D 124.
	Software de programación: WeMake (solo windows), Mobile APP for Arduino Beginner.

Kit Elegoo Robot Arduino Completo Smart Car Autonomo V3.0 	**Componentes:** motor driver board V2.0, sensor ultrasónico, servo motor, módulo de seguimiento de línea, control remoto.
	Conectividad: módulo bluetooth, receptor IR.
	Placa base: Arduino UNO R3.
	Precio: U$D 193.
	Software de programación: Arduino IDE, S4A (scratch for arduino)

RobotGroup N6-Max 	**Componentes:** dos sensores cny70, sensor de ultrasónico, sensores LDR, control remoto, shiel de expansión.
	Conectividad: sensor de control remoto, módulo bluetooth, puerto serie (USART).
	Placa base: DuinoBot v2.3.
	Precio: U$D 336.
	Software de programación: Programable con MiniBloq (similar a scratch) y DuinoPack.

	De forma nativa stand-alone en C/C++, Bitlash y otros lenguajes de alto nivel.

3.5.2 Niveles iniciales

También existen robots que se adaptan a los niveles iniciales siguiendo la filosofía inicial de la enseñanza de la programación. Si bien no se relacionan directamente con el proyecto es importante destacarlos debido al gran impacto que poseen en los niños. A continuación se listan algunos de ellos:

RobotGroup Duino	Es un robot prefabricado por la empresa RobotGroup, pensado para niños desde los tres años. Utilizando el pensamiento lógico y soluciones creativas se programó con tarjetas inteligentes tales como avanzar, girar a la izquierda, girar a la derecha, luz azul, luz verde, luz roja, resetear, pausar, etc.
Beebot	Introduce a los más pequeños al pensamiento computacional a través de diferentes superficies donde el robot debe desplazarse y obliga a los niños a cumplir diferentes misiones.
Pleo	Es un robot con forma de dinosaurio. Pensado para niños que inician el ciclo escolar primario, promueve la experiencia y el aprendizaje. Su lógica se basa principalmente en la respuesta e interacción que tenga el usuario al cuidarlo. Posee un propio sistema operativo llamado Life OS para que los desarrolladores mejoren su funcionamiento a través de programas y comandos.
Cubbeto	Se trata de un robot de madera pensado para niños mayores de tres años, que enseña programación de manera agradable mediante el juego sensorial.

Sobre la base de esta selección de robots programables que existen en el mercado, se optó por los dos que presentan mayor documentación y diversidad en lo que se refiere

a lenguajes de programación, ellos son los que poseen placas **Arduino** y **Lego MindStorms**.

3.5.3 ¿Qué es arduino?

Arduino es una plataforma para realizar proyectos electrónicos que contempla una serie de dispositivos de hardware libre, posee un microcontrolador y una variedad de lenguajes de programación disponibles para realizar programas con instrucciones recibidas por el dispositivo. La lógica de Arduino es usar hardware y software libre, fáciles de manipular, donde se desarrollen e implementen infinitas creaciones por parte de los usuarios a través de una amplia variedad de sensores que se pueden conectar.

Al ser una plataforma de hardware abierto y tener una comunidad tan numerosa facilita la programación del microcontrolador dotándolo de mayor robustez y solidez. Las posibilidades de creación de proyectos son casi infinitas debido a que la mayoría de las variables físicas, por ejemplo, velocidad, aceleración, inclinación, temperatura, fuerza, presión, etc., se pueden convertir en electricidad que Arduino puede medir y utilizar en todo tipo de proyectos.

3.5.3.1 Historia y tipos de placas

Las primeras placas de Arduino fueron creadas en 2005 y contenían un microcontrolador Atmel 8bit Mega AVR. A medida que fueron pasando los años se las fue dotando de mayor capacidad de procesamiento y de entradas para conectar una variedad de dispositivos.

Actualmente existe una diversidad mayor de estas placas, sean oficiales o compatibles, diferenciándose entre ellas principalmente por el precio, la velocidad de procesamiento del microcontrolador y la cantidad de componentes que se le pueden conectar.
Las placas de Arduino que se usan para la mayoría de los proyectos son las siguientes:

- Arduino UNO
- Arduino TRE
- Arduino/Genuino 101
- Arduino Zero
- Arduino Zero Pro
- Arduino Yun
- Arduino Leonardo
- Arduino Due
- Arduino Mega
- Arduino Mega ADK

Entre las ventajas del uso de Arduino respecto a otras plataformas, encontramos las que se mencionan a continuación:

- simplifica el trabajo con microcontroladores,
- económico,
- multiplataforma,
- entorno de programación sencillo,
- software libre y extensible mediante librerías en C++,
- hardware libre y extensible.

3.5.3.2 Lenguajes de programación y entornos de desarrollo

Los proyectos programados en las placas Arduino son llamados skecth. Estos se cargan y ejecutan en la placa. Entre los entornos de desarrollo más conocidos para su programación se destacan el Arduino Software IDE y el IDE Atmel Studio.

Debido al gran éxito que han tenido estas placas, se han desarrollado diferentes lenguajes de programación de alto nivel totalmente compatible con su plataforma. Algunos de ellos son:

- **ArduBlock:** es un lenguaje de programación gráfica que contiene bloques con instrucciones, muy similar a scratch.
- **Snap4Arduino:** se trata de otro lenguaje de programación visual que permite interactuar sin problemas con casi todas las placas Arduino. Este lenguaje no compila el código, pero se comunica con Arduino mientras está conectado a la computadora.
- **C#:** utilizando la librería CmdMessenger se logra conectar con la placa Arduino para mandar y recibir comandos.
- **Python:** utilizando librerías propias de python es posible conectarse al Arduino a través del puerto serial.
- **Tiny Basic para Arduino:** consiste en una versión muy sencilla y simplificada de un intérprete del lenguaje de programación BASIC
- **S4A:** se refiere a una modificación de scratch para programar de forma sencilla arrastrando y soltando bloques para tratar con sensores y actuadores conectados a una placa Arduino.

3.5.4 ¿Qué es Lego Mindstorms?

Lego Mindstorms es una serie de robots programables diseñados para la enseñanza a través de la robótica educativa dotados de diferentes kits que, gracias a la flexibilidad de Lego, permiten a al estudiante diseñar, construir, programar y controlar sus propios robots.(Osorio J. A. C., Salazar O. A. y Mendoza J. A., 2009) El cerebro de este tipo de robots es un bloque programable que posee entradas y salidas para controlar todos sus sensores y motores, además de ofrecer la posibilidad de incluir comunicación inalámbrica a través de bluetooth y/o wifi mediante un dispositivo USB.

3.5.4.1 Historia y tipos de robots

En 1998 la empresa Lego decidió dar un salto importante en la robótica al crear su primer robot comercial: Robotic Invention System. Esa empresa y el Massachusetts

Institute of Technology (MIT)[15] llegaron a un acuerdo para crear la línea de Lego Mindstorms, su resultado fue el MIT Programmable Brick (bloque programable) que se usa actualmente en los nuevos modelos de robot. El bloque programable que venía en el set contaba con tres conectores para capturar información de diversos sensores y dispositivos electrónicos, también, permitía almacenar hasta cinco programas en su memoria interna. La comunicación con la computadora se realizaba a través de un puerto infrarrojo que permitía transferir el firmware y los programas a ejecutar.

En el año 2006 se lanzó al mercado el robot Lego Mindstorms NXT, como en la versión RCX, contó con dos versiones para la venta, una comercial y otra educativa. Las diferencias con su predecesor radicaron principalmente en las capacidades de procesamiento del microcontrolador. Poco tiempo después se lanzó la versión RCX 2.0, la cual mejoró notablemente la comunicación debido a que se le agregaron dos interfaces, USB y bluetooth, para comunicarse con computadoras y con otros bloques, celulares, tablets y más.

La última generación de Lego fue la innovadora Lego Mindstorms EV3, se lanzó al mercado en 2013. La diferencia más notable respecto a sus versiones anteriores radica en las modificaciones tecnológicas del bloque principal programable o cerebro del robot. Además de mejorar su velocidad y capacidad de procesamiento incorporó un slot para incluir una memoria micro SD de hasta 32 gigabytes, donde se puede agregar un sistema operativo para el arranque. En este modelo, al igual que en los anteriores, existen dos versiones, una comercial y otra educativa. La diferencia entre ellas radica especialmente en el precio y en los componentes de cada set.

Figura 3.2: Evolución de Lego Mindstorms

Abajo se presenta un cuadro comparativo entre los tres modelos de Lego.

	RCX	NXT	EV3
Fecha de	1998	julio 2006	septiembre 2013

[15] "MIT - Massachusetts Institute of Technology." [En línea]. Disponible en: http://web.mit.edu/. [Último acceso: Junio 2018].

lanzamiento			
Pantalla	segmentada LCD monocromo	100×64 pixel LCD monocromo	178×128 pixel LCD monocromo
Procesador	Hitachi H8/300 @16 MHz Atmel AT91SAM7S256	(ARM7TDMI core) @48 MHz TI Sitara AM1808	(ARM926EJ-S core) @300 MHz
Memoria principal	32 KB RAM 16 KB ROM	64 KB RAM 256 KB Flash	64 MB RAM 16 MB Flash
Memoria expansible	-	-	Ranura microSD
Puerto USB	-	-	Sí
Wifi	-	-	adaptador opcional a través del puerto USB
Bluetooth	-	Si	Sí
Principales lenguajes para programar	NQC NXC	Lenguaje gráfico NXT-G NXC	LEGO Mindstorms EV3 Software

Tabla 3.2: Comparativa de los robots Lego Mindstorms

3.5.4.2 Lenguajes de programación

Desde el lanzamiento del robot Lego Mindstorms EV3, en septiembre de 2013 hasta la actualidad, fueron desarrollados muchos lenguajes de programación y entornos de desarrollo. El software oficial que brinda Lego para programar sus robots EV3 se llama EV3 Programmer. Se trata de un lenguaje gráfico que puede ejecutarse en Windows, Mac y también en celulares con Iphone y Android, lo que permite controlar a todos los sensores y dispositivos que se le conecten.

A continuación se listan algunos lenguajes no oficiales de Lego que permiten controlar al robot con todos sus componentes.

- **EV3Basic**, Microsoft Small Basic: es un compilador de Basic orientado al EV3 ejecutable solo en Windows.
- **ev3_scratch**: se trata de un lenguaje de programación visual muy conocido que posee complementos que permiten trabajar con EV3. Desde el programa se pueden mandar órdenes a través de bluetooth.
- **LabVIEW**: consiste en un entorno de programación visual, el cual, a través de módulos que se instalan, puede hacer uso de los componentes de los robots Lego. Tiene documentación para la versión NXT, pero es deficiente aún para los robots EV3.

- **.NET**: contiene una API para ser utilizada con Windows y controlar todos los sensores del EV3, se puede conectar a través de bluetooth, wifi o USB.
- **LeJOS**: radica en un lenguaje de alto nivel efectuado en la plataforma Java.
- **BricksOS**: se basa en un lenguaje de alto nivel realizado en C++.
- **RobotC**: consta de un lenguaje pago que posee su propio entorno de programación y utiliza como base el lenguaje C. Incluye una buena herramienta de depuración.
- **ev3dev**: se apoya en un sistema operativo Debian basado en Linux. Permite controlar todos los sensores y motores del Lego; además es compatible con muchos lenguajes de scripting. El sistema operativo se instala en la microSD del robot sin modificar el firmware original del EV3.
- **MATLAB**: consta de una multiplataforma y ofrece un entorno de desarrollo con su lenguaje de programación M. Mediante una extensión gratuita proporciona soporte para los robots EV3.

Capítulo 4 - Integración de la robótica educativa en DaVinci-C

Uno de los objetivos principales de esta tesis era realizar la integración de un robot físico programable con la aplicación, ya existente, DaVinci Concurrente. En este capítulo se presentará el modo de implementación de dicho objetivo.

Se desarrolló un intérprete que reconoce un conjunto limitado de primitivas del lenguaje de programación DaVinci-C, el cual brinda las instrucciones necesarias al robot físico para que las ejecute, y su comportamiento se vea reflejado en una ciudad física.

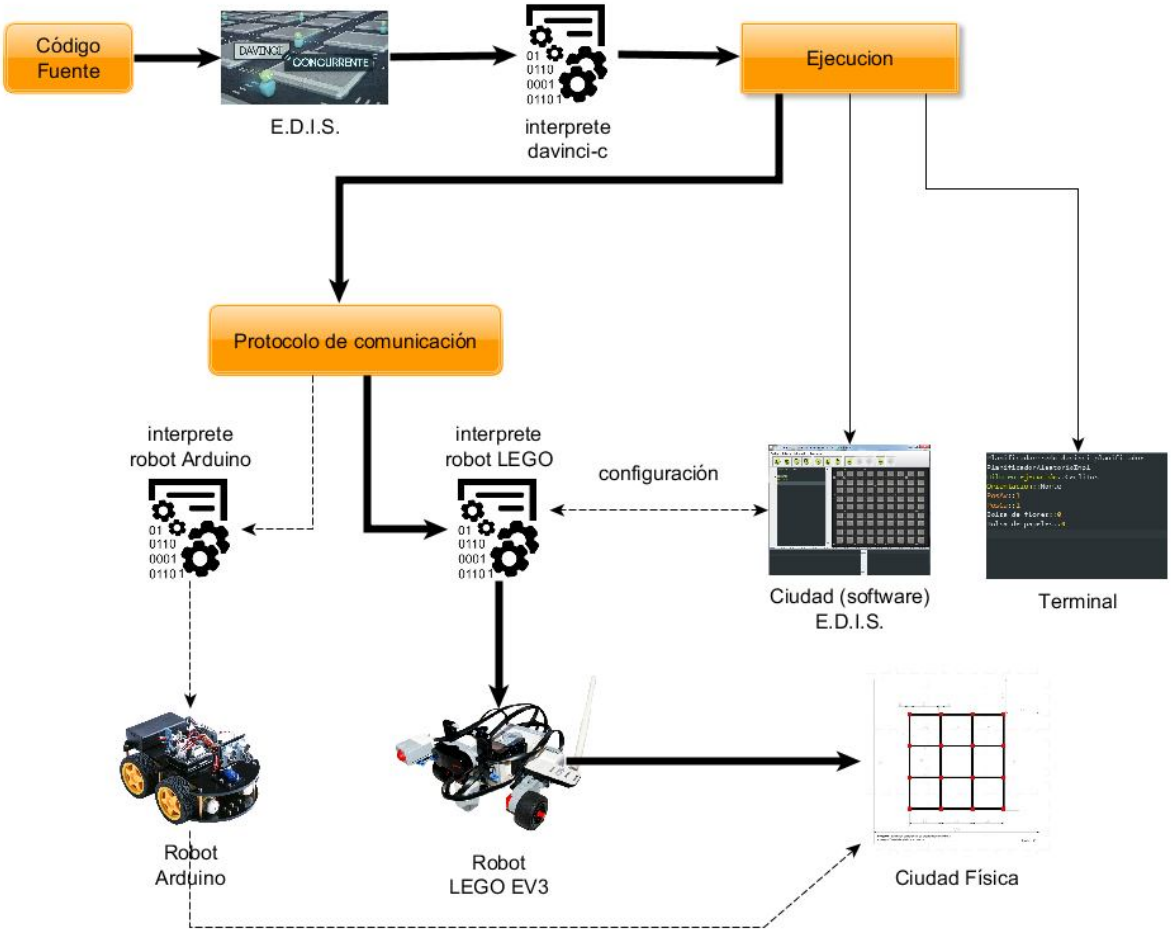


Figura 4.1: Esquema general del proyecto

A través de la robótica educativa se intenta despertar, en los estudiantes, la motivación necesaria para ingresar al mundo de la programación. La comunicación que posee el

intérprete DaVinci-C con el nuevo intérprete del robot es mediante una arquitectura cliente servidor, tal cual se lo muestra en la siguiente imagen ilustrativa.

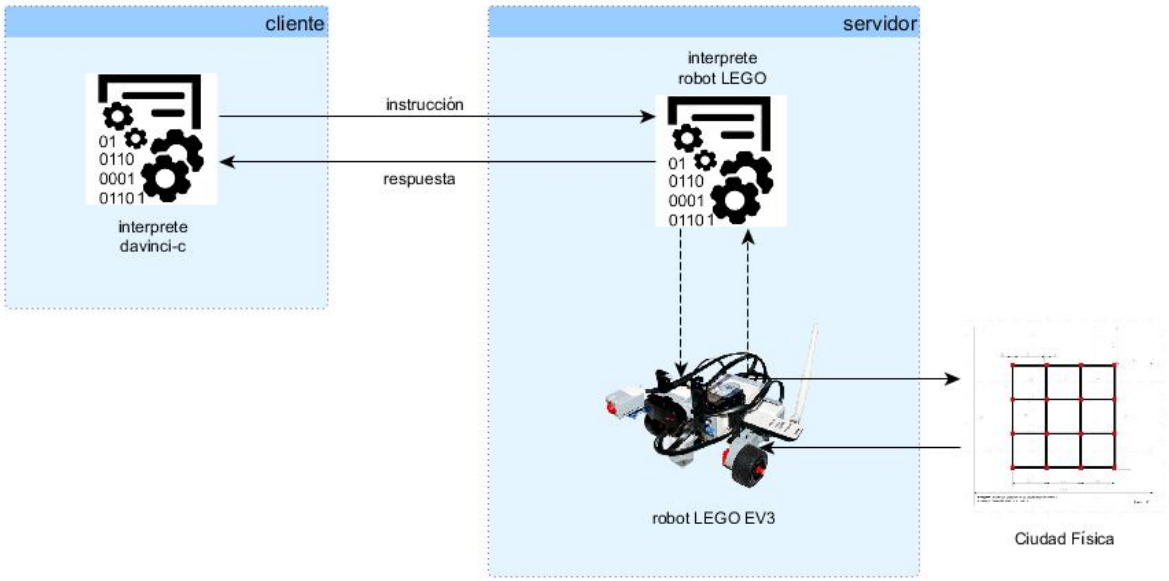


Figura 4.2: Arquitectura cliente-servidor general

4.1 Problemas con los actuales lenguajes y entornos de desarrollo

Si bien la plataforma Arduino es mucho más económica que la de Lego Mindstorms, la orientación del mismo no se relaciona directamente con lo apuntado en esta tesis. Dado que su enfoque requiere que los estudiantes posean saberes previos en otras disciplinas como la electrónica, mecánica, física, programación, etc. Los robots Lego Mindstorms siguen la filosofía de su marca: se puede armar y construir cualquier clase de objeto con el solo hecho de unir bloques interconectables, por ende, su aprendizaje es relativamente más sencillo.

El problema principal que tienen los actuales entornos de desarrollo y lenguajes de programación, tanto en Arduino como en Lego, es que no se concentran en la orientación del aprendizaje de la programación inicial, sino que lo hacen en la manipulación del robot con sus componentes y sensores.

El objetivo de esta tesis es manipular el funcionamiento del robot para que pueda reconocer un conjunto de instrucciones dadas por el intérprete DaVinci-C, a partir de un código fuente programado por el estudiante, y este pueda ejecutarlas de manera física.

4.2 Protocolo de comunicación entre el lenguaje y el robot

Se fijó un conjunto de reglas que permiten establecer la comunicación entre el intérprete y el robot. El resultado es la reproducción del comportamiento del robot, visualizado en la pantalla, en un espacio físico.

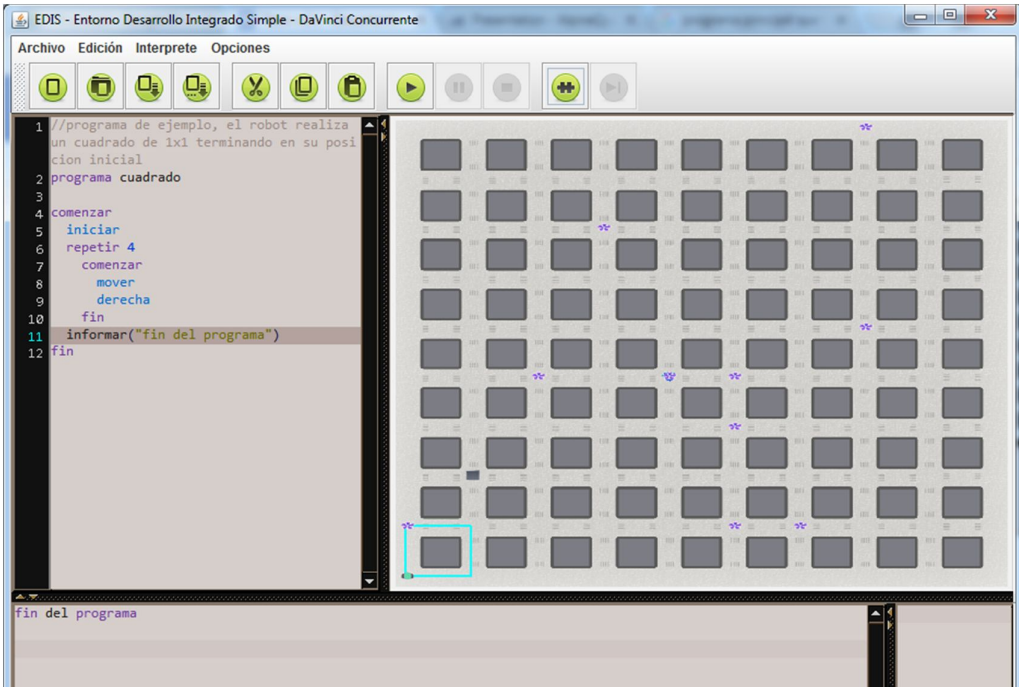


Figura 4.3: E.D.I.S. (Entorno de Desarrollo Integrado Simple)

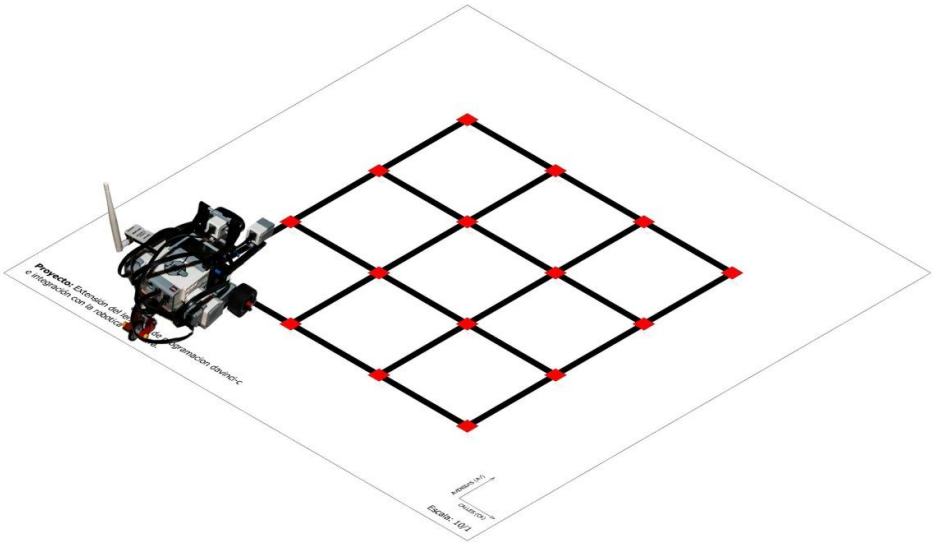


Figura 4.4: Maquetado de la ciudad física

4.2.1 Palabras reservadas del lenguaje de programación DaVinci-C

El lenguaje de programación DaVinci-C contiene declaraciones léxicas que poseen un comportamiento asociado a ellas, formando así la semántica del lenguaje. Se pueden categorizar de la siguiente manera:

Declaraciones	programa, subprogramas, procedimiento, en, sa, es, hilos, hilo, variables, numero, logico, texto, caracter, semaforoBinario, semaforoGeneral, comenzar, fin
Estructuras de control	repetir, mientras, si, sino
Primitivas	iniciar, mover, derecha, tomarFlor, depositarFlor, tomarPapel, depositarPapel
Funciones	posAv, posCa, hayFlorEnLaEsquina, hayFlorEnLaBolsa, hayPapelEnLaEsquina, hayPapelEnLaBolsa, hayObstaculo, hayRobot, longuitud, sustraer, numeroATexto, textoANumero, logicoATexto
Procedimientos	aleatorio, informar, pos, pedir, iniciarSemaforo, arrancar, avisar, esperar

Tabla 4.1: Palabras reservadas de davini-c

4.2.2 Subconjunto de primitivas

De todas las declaraciones léxicas que contiene el lenguaje, existe un conjunto de ellas que se pueden implementar con un robot físico, y son las siguientes:

iniciar, mover, derecha, tomarflor, tomarpapel, depositarflor, depositarpapel, hayFlorEnLaBolsa, hayPapelEnLaBolsa, hayFlorEnLaEsquina, hayPapelEnLaEsquina, hayObstaculo, hayRobot, longuitud, sustraer, numeroATexto, textoANumero, logicoATexto, aleatorio, informar, pedir, iniciarSemaforo, arrancar, avisar, esperar, pos, programa, subprogramas, procedimiento, en, sa, es, hilos, hilo, variables, numero, logico, texto, caracter, semaforoBinario, semaforoGeneral, comenzar, fin, repetir, mientras, si, sino

Muchas de ellas tienen incidencia directa con el robot físico y otras son interpretadas por el lenguaje, no teniendo una incidencia directa con aquel. Las primitivas que poseen incidencia directa con el robot, es decir, las que forman parte de la interacción con él son las que se mencionan a continuación:

iniciar, mover, derecha, tomarflor, tomarpapel, depositarflor, depositarpapel,

hayObstaculo, hayRobot, posAv, posCa,

El resto de las primitivas no inciden directamente con el robot, pero aún así son interpretadas y pueden usarse en los programas desarrollados por el estudiante en el lenguaje DaVinci-C y visualizar su efecto sobre la ciudad física. Es decir, tanto las estructuras de control, subprogramas (funciones y procedimientos) y el resto de las declaraciones léxicas se pueden utilizar en los algoritmos que desarrolle el alumno, entendiéndose de que las mismas son primitivas ejecutadas por el intérprete DaVinci-C

Por una cuestión de recursos y de tiempo, en esta primera etapa solo se implementó un subconjunto reducido de las primitivas que van a tener incidencia directa con el robot. Ellas se nombran seguidamente:

iniciar, mover, derecha, hayObstaculo

En esta etapa se dará soporte para un solo robot en la ciudad, sin descartar la posibilidad de que en un futuro se puedan agregar más realizando las modificaciones necesarias, e implementando más primitivas que incidan en el robot físico.

Se destaca que con las primitivas implementadas, los alumnos podrán aprender todas las estructuras de control y visualizar las diferentes acciones que desarrollará el robot en la ciudad física. Así como también, combinar su uso dentro del cuerpo de subprogramas, comprendiendo de esta manera los conceptos básicos y fundamentales para el aprendizaje inicial de la programación.

4.2.3 Primitivas del robot

Se realizó un estudio de factibilidad tanto para las plataformas de Lego Mindstorms como para las de Arduino, obteniendo resultados interesantes a tener en cuenta ante una posible implementación. Las primitivas consideradas para su estudio fueron elegidas sobre la base del subconjunto previamente realizado en función a su incidencia directa en el robot. A continuación se detallan cada una de ellas:

Primitiva: INICIAR

Semántica:

Posicionarse en la esquina inicial de la ciudad.

Implementación:

Inicialización de los componentes internos del robot y su comunicación. La colocación de la ubicación física en la ciudad es manual e inicialmente tendría que estar orientado al norte.

Primitiva: MOVER

Semántica:

El robot avanza una cuadra en la dirección en la que se encuentre.

Implementación:

Tanto en Lego EV3 como en Arduino, el problema es la precisión de los motores al medir el recorrido realizado. Una opción es contar la cantidad de vueltas de

las ruedas, pero esto sería impreciso y dependería mucho de la superficie en la cual se esté movilizando.

Algunas de las alternativas para ambos modelos serían las siguientes:

- A través de sensores de seguimiento de línea, por ejemplo, una línea negra, hasta llegar a un color diferente detectado por un sensor de color indicando la esquina correspondiente.
- Si la ciudad estuviese rodeada de paredes se podrían utilizar sensores de ultrasónico (conociendo la distancia entre cada cuadra) que midan la distancia en llegar desde un punto a otro.

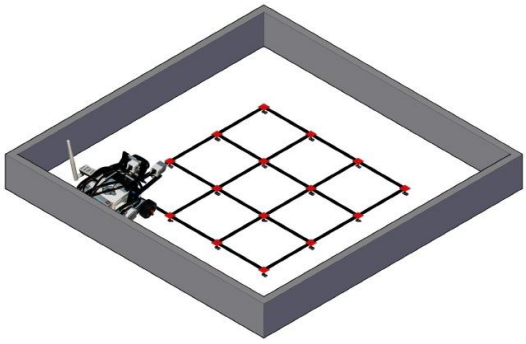


Figura 4.5: Ejemplo de una posible implementación para la primitiva MOVER

Primitiva: DERECHA

Semántica:

El robot debe girar 90 grados en sentido horario.

Implementación:

Con un giroscopio se pueden medir los grados de rotación que realiza el robot en la dirección que se designe.

- ***En Lego EV3:*** en este robot existen tres posibilidades; una, con ayuda de un giroscopio; otra, haciendo uso de los motores y contabilizando el tiempo de funcionamiento de una rueda, de esta manera se podría realizar la rotación; y la tercera es a través del uso de otros sensores. En esta última se utilizarían marcas de colores en las esquinas junto con el sensor de detección de colores que permitan girar el robot hasta la próxima marca.
- ***En Arduino UNO:*** se pueden utilizar las mismas posibilidades que en Lego EV3; y también obtener ruedas direccionales que giren sobre su propio eje facilitando el giro de 90 grados.

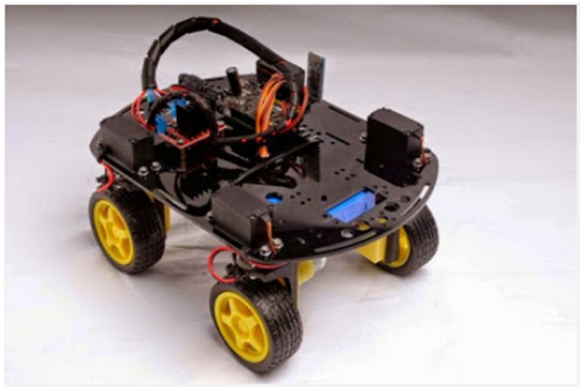


Figura 4.6: Ejemplo de robot con ruedas direccionales - "Andromina ON ROAD"

Primitivas: TOMAR FLOR / TOMAR PAPEL

Semántica:

El robot recoge una flor/papel en la esquina que se encuentre y la incluye en una bolsa virtual; si no existiera la flor/papel en la esquina se producirá un error que abortará la ejecución programa.

Implementación:

Integrando una ciudad inteligente que posea leds de diferentes colores (para las flores y los papeles) en las esquinas, cuando el robot reciba la instrucción de tomar flor/papel las detectará con sensores de color conectados a ellos. Una vez recolectadas las flores/papeles, los leds de la ciudad se apagarán. Instantáneamente se visualizará la cantidad de flores/papeles que se van recogiendo a través de un display integrado en el robot. La ciudad estará conectada y se comunicará con el robot.

Primitivas: DEPOSITAR FLOR / DEPOSITAR PAPEL

Semántica:

Deposita una flor/papel en la esquina que se encuentra el robot. Si se intentase depositar una flor en una esquina y el robot no tuviera flores/papeles en la bolsa se producirá un error que abortará la ejecución del programa.

Implementación:

Al igual que en las primitivas anteriores (tomar flor/tomar papel), la ciudad deberá tener un sistema de control para el apagado y encendido de leds que representará a las flores/papeles.

Primitivas: HAY FLOR EN LA BOLSA / HAY PAPEL EN LA BOLSA

Semántica:

Retorna verdadero en caso de existir en la bolsa del robot al menos una flor.

Implementación:

Tanto para Arduino como para LEGO EV3, a través de un led indicador, se puede informar si existe flor/papel en el bolsa virtual del robot a través del cambio de color y/o porque titila.

Primitivas: HAY FLOR EN LA ESQUINA / HAY PAPEL EN LA ESQUINA

Semántica:

Retorna verdadero en caso de existir al menos una flor en la esquina en la que se encuentra el robot; en caso contrario retorna falso.

Implementación:

Al igual que las primitivas anteriores, contando con una ciudad dotada de leds manipulables en modo ejecución y robots con sensores de color, se podría detectar fácilmente cuando hay una flor/papel (representado en la ciudad con leds de colores diferentes) en una esquina determinada.

Primitiva: HAY OBSTACULO

Semántica:

Retorna verdadero si en la próxima esquina, en la dirección en la cual se encuentra orientado el robot, hay un obstáculo.

Implementación.

Tanto para Arduino como para Lego EV3, se puede utilizar el sensor de ultrasónico para realizar la detección. Por lo general, tiene un alcance de 3 a 250 centímetros. Como los obstáculos a detectar son solo los de la próxima esquina, debería tener en cuenta la distancia con una mínima tolerancia.

Primitiva: HAY ROBOT

Semántica:

Retorna verdadero si en la próxima esquina, en la dirección en la cual se encuentra orientado el robot hay otro robot.

Implementación:

El único problema que existiría sería la forma que tendría el robot 1 para analizar si el objeto de la esquina siguiente es otro robot o un obstáculo. Una posibilidad para solucionar ese problema, tanto en Arduino como en Lego EV3, sería que el robot 1 analice con determinados sensores si existe un objeto o no en la próxima esquina y distinguir los obstáculos con un determinado color.

Primitiva: POSAV

Semántica:

Obtener el número de avenida en el que se encuentra.

Implementación:

El intérprete DaVinci-C va guardando la posición en la que se encuentra el robot, tanto para Arduino como para LEGO EV3, pudiéndose mostrar el dato solicitado mediante un display.

Primitiva: POSCA

Semántica:

Obtener el número de calle en el que se encuentra.

Implementación:

El intérprete DaVinci-C va guardando la posición en la que se encuentra el robot, tanto en Arduino como en Lego EV3 se puede mostrar el dato solicitado a través de un display.

4.2.4 Protocolo

El canal de comunicación entre el intérprete y los robots es un socket que escucha en puerto 2048 con una arquitectura del tipo cliente-servidor, utiliza TCP como capa de transporte e IP como capa de red (usando wifi). Cada mensaje es codificado en una cadena de caracteres y siempre hay un mensaje de retorno en función a lo solicitado.

A continuación se detallan las primitivas que tienen incidencia directa con el robot, como así también, los retornos que deben tener en caso de éxito y/o error.

Mensaje	Retornos	Descripción
INICIAR	OK	Los componentes se iniciaron correctamente.
	ERROR	Falla de algún componente mal iniciado.
MOVER	OK	La instrucción se ejecutó correctamente.
	TTPRES	El sensor táctil (botón de pánico) está siendo o fue presionado.
	ERROR	Falla de algún motor.
DERECHA	OK	La instrucción se ejecutó correctamente.
	TTPRES	El sensor táctil (botón de pánico) está siendo presionado.
	ERROR	Falla del sistema de rotación y/o giroscopio.
TOMARFLOR / TOMARPAPEL	OK	La instrucción se ejecutó correctamente.
	ERROR	No existe flor /papel en la esquina.
DEPOSITARFLOR / DEPOSITARPAPEL	OK	La instrucción se ejecutó correctamente.
	ERROR	No tiene flor/papel en la bolsa.
HAYFLORENLAESQUINA / HAYPAPELENLAESQUINA	OK	La instrucción se ejecutó correctamente.
HAYROBOT / HAYOBSTACULO	OK	La instrucción se ejecutó correctamente.
	TTPRES	El sensor táctil (botón de pánico) está siendo presionado.

	ERROR	Falla en el sensor de ultrasónico.
POSAV / POSCA	OK	La instrucción se ejecutó correctamente.

Tabla 4.2: Protocolo de comunicación

Como se mencionó anteriormente, en esta etapa solo se dará soporte a las siguientes primitivas:

- **iniciar**
- **mover**
- **derecha**
- **hayObstaculo**

4.3 Elección de las herramientas utilizadas

El uso de la plataforma Arduino requiere que los estudiantes tengan incorporados conceptos básicos de electrónica y otras disciplinas que no forman parte del objetivo principal de esta tesis. La idea es abstraer a los estudiantes de esos problemas y usar la robótica educativa como motivación para integrar los conocimientos básicos en la programación. Si bien la plataforma Lego tiene un costo mayor que la Arduino, es mucho más amigable al momento de ensamblar los proyectos y dejarlos listos para su programación.

Estas son las razones por las que se decidió utilizar el robot Lego Mindstorms en su versión más actual, EV3 (Evolution 3). Se implementó lo necesario para que reconozca las primitivas enviadas por el intérprete DaVinci-C del lado del cliente.

De todos los lenguajes de programación y entornos de desarrollo existentes, para la programación de los robots Lego Mindstorms, se eligió el sistema desarrollado en el proyecto “EV3DEV” (Hempel R. y Lechner D., 2015) debido a la versatilidad, documentación y compatibilidad que posee. Se trata de un sistema operativo Linux que, instalado en la memoria microSD del robot, puede programarse utilizando diversos lenguajes de programación que reconoce el robot y son mantenidos por la comunidad que hace uso de ellos. Se pueden citar los siguientes como ejemplo:

- Python
- JavaScript
- Java
- Java (LeJOS-compatible)
- Go
- C++
- C
- Vala

La ejecución de estos lenguajes de programación más avanzados permite que el robot EV3 realice acciones que con el sistema estándar del robot “EV3 Programmer” no serían posibles.

Python fue el seleccionado para la implementación del intérprete del robot debido a su sencillez, rapidez, flexibilidad y demás características, que lo hacen un lenguaje ideal para este tipo de sistemas, sin contar que es interpretado y multiparadigma.

4.4 Comunicación entre el intérprete y el robot

Si bien DaVinci-C fue desarrollado en Java, por una cuestión de simplicidad, se decidió que el nuevo intérprete del robot lo fuera en el lenguaje de programación Python.

El esquema general, respetando la arquitectura cliente servidor, se definió así:

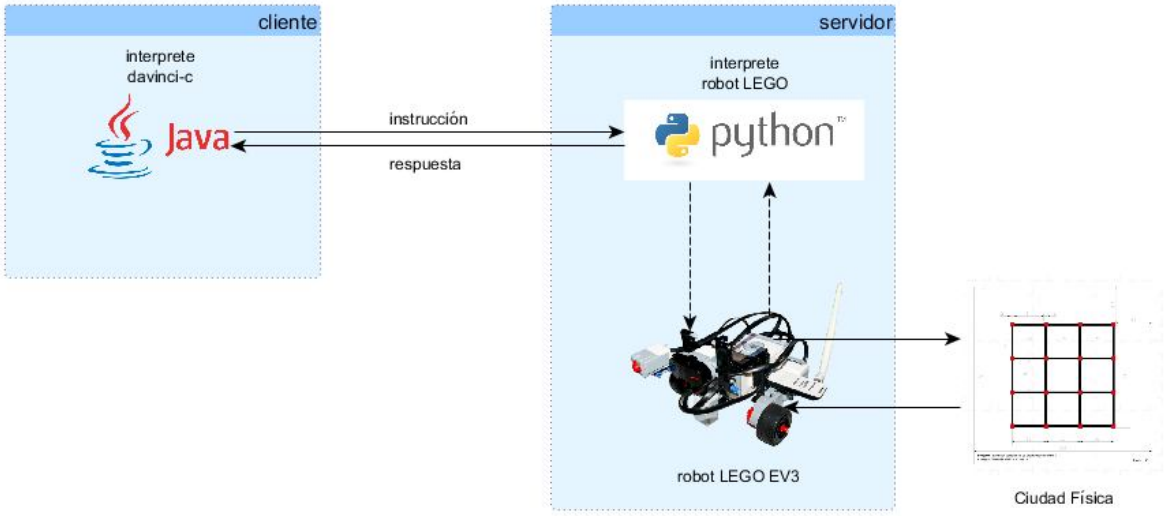


Figura 4.7: Comunicación de los intérpretes

4.5 Recursos limitados

Al no contar con los recursos necesarios del robot Lego EV3, el intérprete sólo se limita a reconocer ciertas primitivas del lenguaje; las suficientes para reconocer algoritmos sencillos en una programación inicial en los cuales el estudiante puede ver reflejado el comportamiento de un robot en una ciudad física. Las primitivas que va a reconocer el intérprete desarrollado en el robot son las siguientes:

- iniciar
- mover
- derecha
- hayObstaculo

Como se mencionó en el apartado 4.2.3 - Primitivas del robot, que el intérprete reconozca este conjunto limitado de primitivas no significa que se deben omitir en el código de los programas, otras primitivas que puede reconocer el intérprete DaVinci-C, entre ellas todas las estructuras de control y subprogramas.

4.6 Especificación e implementación del intérprete

4.6.1 Requerimientos

Se estableció un conjunto de requerimientos para la integración correspondiente, muchos de ellos se debieron a la falta de recursos y tiempos de desarrollo.

- Habrá un solo robot en la ciudad
- Si está habilitada la ciudad física, los obstáculos físicos tendrán prioridad en la ejecución del algoritmo desarrollado, independientemente de que existan obstáculos virtuales en la ciudad del EDIS.
- Tanto el cliente (desarrollo local) como el robot físico, deben contar con una canal de comunicación bidireccional.
- No se podrán recoger papeles y flores tangibles debido a las limitaciones físicas del robot
- Primitivas con incidencia directa en el robot excluidas:
    - hayRobot, tomarpapel, depositarflor, depositarpapel, hayFlorEnLaBolsa, hayPapelEnLaBolsa, hayFlorEnLaEsquina, hayPapelEnLaEsquina, pos.

4.6.2 Diagrama de clases

A continuación se muestra el diagrama de clases correspondiente al intérprete implementado en el robot.

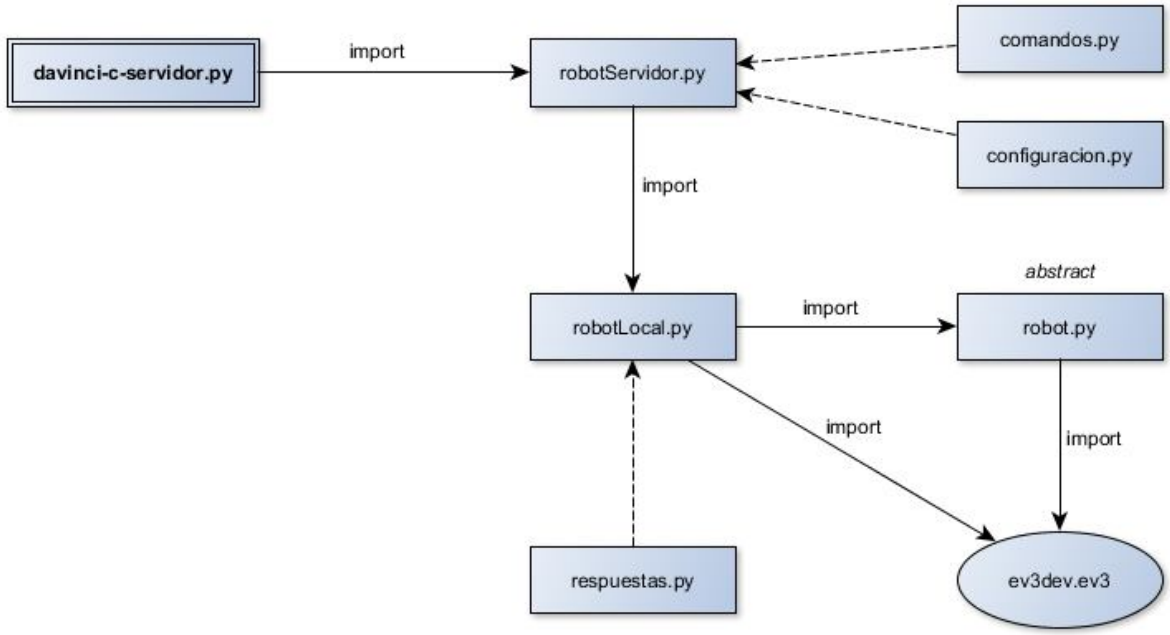


Figura 4.8: Diagrama de clases del intérprete del robot

donde,

davinci-c-servidor.py: es la clase principal para iniciar el servidor en el robot.

robotServidor.py: posee los métodos para establecer una conexión vía sockets, tiene la lógica para recibir comandos de instrucciones y enviar respuestas de acuerdo a lo solicitado.

robotLocal.py: implementación de las instrucciones que recibe el robot.

robot.py: posee los métodos abstractos de las instrucciones que puede recibir el robot.

respuestas.py: constantes de las posibles respuestas que puede emitir el servidor del robot.

comandos.py: constantes de las instrucciones que recibe el robot.

configuracion.py: constantes de las variables para la configuración del servidor en el robot.

ev3dev.ev3: interfaz para controlar los componentes conectados al robot.

4.6.3 Definición de la implementación

La implementación del nuevo intérprete está programada en Python utilizando los módulos e interfaces que provee EV3DEV, en el siguiente cuadro se detallan las primitivas y su descripción.

Instrucción	Descripción
iniciar	Inicialización de los componentes internos del robot y su comunicación. La colocación en la ubicación física en la ciudad es manual e inicialmente tendría que estar orientado al norte.
mover	El robot debe avanzar una cuadra en la dirección que se encuentre. Para realizar esta instrucción utiliza el sensor de color para seguir la línea negra dibujada en la ciudad hasta llegar a completar la distancia previamente configurada en el intérprete del robot.
derecha	El robot debe girar 90 grados en sentido horario. Para realizar esta instrucción se usan los motores principales ejecutando su funcionamiento, uno adelante, y el otro en sentido inverso, obteniendo así la rotación correspondiente. Debido a que las ruedas no giran sobre su eje, antes de iniciar la rotación el robot debe avanzar, y luego de la rotación, retroceder. De esta manera se corrige la desviación que produce la rotación del robot.
hayObstaculo	El robot debe poder reconocer si en la próxima esquina existe un obstáculo que le impida continuar su movimiento. Para esto se usa el sensor infrarrojo que corrobora si existe o no un objeto sólido a una distancia menor a un valor previamente configurado en el intérprete del robot.
terminar	Se envía una señal de finalización de programa al robot para que detenga correctamente todos los dispositivos conectados. Esta instrucción se envía cuando el programador cierra el entorno de desarrollo.

Tabla 4.3: Definición de la implementación del robot

En el Anexo I, Código fuente intérprete robot, se describe la implementación con las instrucciones descriptas en la tabla de arriba.

4.7 Especificación de la ciudad física

Para realizar las pruebas pertinentes a la implementación del intérprete en el robot Lego EV3 se montó una ciudad física sobre bloques goma eva encastrable, donde se dibujó una ciudad de 3 x 3. Las medidas que se usaron para la mencionada ciudad se detallan en la imágen ilustrativa que sigue.

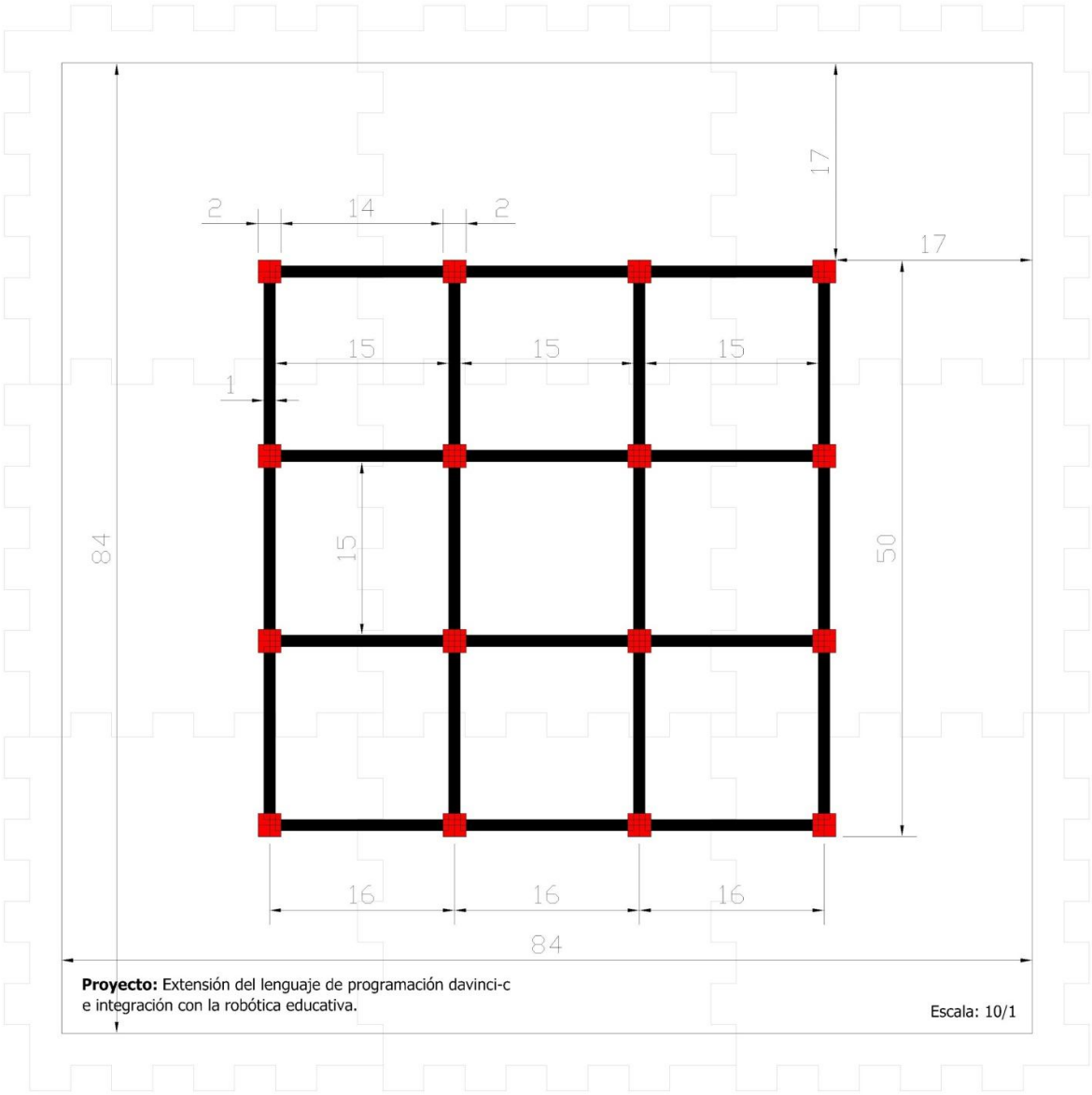


Figura 4.9: Especificación de la ciudad física

Se dejaron marcas en las esquinas de la ciudad para utilizar cuando el robot cuente con otro sensor de color. Estas fueron diferenciadas de las guías que posee el robot para avanzar de cuadra en cuadra, siendo su superficie de 2 cm x 2 cm.

4.7.1 Superficie

Se probaron varias superficies antes de tomar la decisión final para la construcción de la ciudad física, siempre teniendo en cuenta las características de portabilidad y usabilidad. Inicialmente se realizó la ciudad física sobre una tela blackout vinílica, dibujando las líneas con cinta aisladora. Si bien en esta primera opción la tela era bastante transportable, el problema se generó cuando la tracción de las ruedas del

robot, al deslizarse, formaba arrugas y producía desfases considerables en su recorrido.

La segunda opción fue dibujar e imprimir la ciudad en un plotter para que el robot realice sus recorridos. Si bien fue una buena decisión, el problema surgió cuando se la enrollaba para su traslado, se iba degradando, arrugando y perdiendo consistencia. Por este motivo se decidió utilizar bloques de goma eva encastrables, pegando arriba la ciudad impresa. De esta manera se obtuvo la portabilidad y usabilidad que se buscaba.

4.8 Lego Mindstorms EV3

4.8.1 Características principales

El kit de Lego Mindstorms EV3 contiene un bloque programable con un firmware Linux en un microprocesador TI Sitara AM1808 con 64 MB de RAM y la posibilidad de conectarle una tarjeta micro SD como así también un dispositivo USB. Los robots que se pueden armar con el kit del Lego Mindstorms EV3 son muchísimos, aun así desde su sitio oficial ofrece la guía para armar los siguientes:

TRACK3R	SPIK3R	R3PTAR	GRIPP3R
EV3RSTORM	BOBB3E	BANNER PRINT3R	RAC3 TRUCK
DINOR3X	KRAZ3	EV3D4	EL3CTRIC GUITAR
EV3MEG	EV3 GAME	MR. B3AM	ROBODOZ3R

Su bloque programable, o modulo central, posee un puerto USB que sirve de apoyo y le brinda la capacidad de comunicarse por wifi con otros dispositivos, abriendo así un abanico de posibilidades para el diseño y creación proyectos.

Contiene cuatro puertos de entrada para recibir información proveniente de los sensores, y cuatro puertos de salida para accionar a través de motores. Según la documentación de Lego, si el dispositivo no se utiliza con el software propietario de la marca, el robot no reconocerá automáticamente la posición de los sensores a los que se esté conectando, por este motivo se debe respetar la asignación por defecto que viene de fábrica:

- Puerto 1: sensor táctil.
- Puerto 2: giroscopio/sensor de temperatura.
- Puerto 3: sensor de color
- Puerto 4: sensor ultrasónico/sensor infrarrojo.

Pasa exactamente lo mismo con los puertos de salida, la asignación por defecto en el robot EV3 es la siguiente:

- Puerto A: motor mediano.
- Puertos B y C: dos motores grandes.
- Puerto D: motor grande.

Tiene la posibilidad de conectarse con hasta cuatro bloques programables por los puertos USB y programarlos todos como si fueran uno solo, multiplicando de esta manera todas sus características y puertos utilizables.

Al pertenecer a una comunidad tan grande, abre las puertas para que sistemas como el EV3DEV trabajen en él, extendiendo de esta manera los lenguajes de programación para la manipulación del robot diseñado.

4.8.1.1 Componentes del kit

Los componentes que trae en el kit estándar de Lego Mindstorms EV3, son los siguientes:

- 1 x bloque programable EV3,
- 1 x sensor táctil EV3,
- 1 x sensor de color EV3,
- 1 x sensor infrarrojo EV3,
- 2 x servomotores grandes EV3,
- 1 x servomotor mediano EV3,
- 1 x baliza infrarrojo,
- 594 elementos de LEGO Technic para construcción.

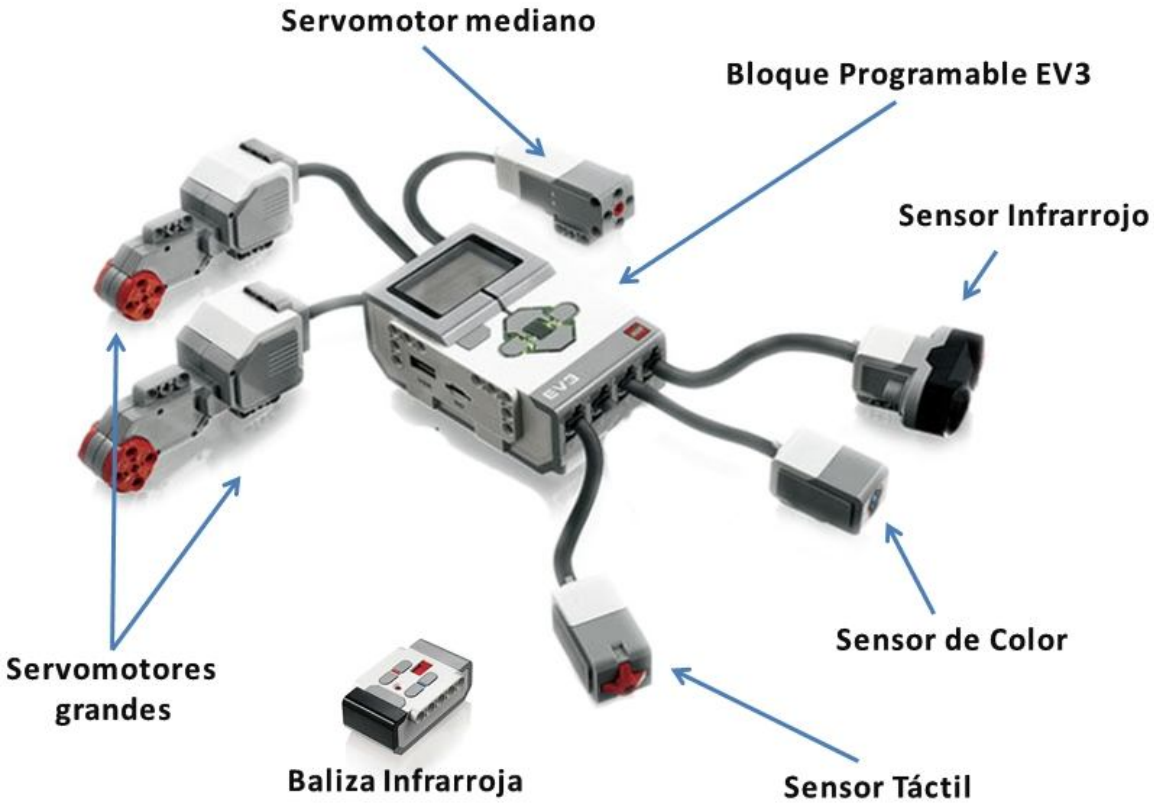


Figura 4.10: Componentes del kit Lego Mindstorms EV3

Para armar el proyecto se utilizó gran parte de los componentes suministrados por el kit. Sin embargo, muchos quedaron almacenados y faltaron otros para hacer más eficiente el funcionamiento del robot. Por ejemplo, con la presencia de otro sensor de luz se podría saber fehacientemente el final de cada esquina en la ciudad.

4.8.2 Integración con EV3DEV

Uno de los lenguajes de programación que soporta EV3DEV es Python, con el cual se programaron las instrucciones del intérprete puesto en el robot. Para la implementación del intérprete del robot se utilizaron diversos componentes suministrados por el kit de Lego Mindstorms, entre ellos se destacan los siguientes:

- 2 x servomotores grandes EV3
- 1 x sensor táctil EV3
- 1 x sensor de color EV3
- 1 x sensor infrarrojo EV3

4.8.2.1 Motores

Los servomotores, o motores inteligentes, funcionan a 160-170 rpm, tienen un sensor de rotación y fueron pensados para ser la base de avance del robot final. La interfaz que brinda EV3DEV (Hempel R., 2015) para el uso de los motores con retroalimentación de posición y direccional, permite un control preciso. Las primitivas que tienen una incidencia directa con el robot para sus movimientos, son las siguientes:

- **run_direct:** ejecuta el motor en un ciclo de trabajo especificado por el atributo "*duty_cycle_sp*". A diferencia de otros comandos, este tendrá efecto inmediatamente.
- **run_forever:** ejecuta el motor hasta que se envíe otro comando.
- **run_timed:** ejecuta el motor durante un tiempo especificado en el atributo "*time_sp*", su tipo de detención es determinada por el atributo "*stop_action*".
- **run_to_abs_pos:** ejecuta el motor hasta una posición absoluta especificada por el atributo "*position_sp*", su tipo de detención es determinada por el parámetro "*stop_action*".
- **run_to_rel_pos:** ejecuta el motor hasta una posición relativa al valor de la posición actual. La nueva posición será la sumatoria de la posición actual más el valor del atributo "*position_sp*". La velocidad se logra configurando el atributo "*speed_sp*". Cuando se alcanza la nueva posición su tipo de detención es determinada por el atributo "*stop_action*".

Los motores se configuran mediante los atributos que se incluyen en las primitivas, influyendo, de esta manera, en el comportamiento del robot radicalmente. A continuación se presenta cada una de ellas:

- **duty_cycle_sp**: establece el punto de ajuste del ciclo de rotación. Las unidades están en porcentaje, los valores admitidos son de -100 a 100. Los valores negativos permiten que el motor gire en inversa.
- **time_sp**: especifica la cantidad de tiempo que el motor funcionará. Las unidades están en milisegundos.
- **position_sp**: ordena al motor moverse el valor del atributo. Las unidades se presentan en porcentaje y van de -100 a 100.
- **speed_sp**: establece la velocidad en grados por segundo, por lo que si se establece speed_sp a 360 el motor va a tratar de girar 360 grados por segundo, o una rotación por segundo.
- **stop_action**: establece la acción al detenerse. Ll valor determina el comportamiento de los motores cuando el comando está configurado para detenerse. Los posibles valores son los siguientes:
    - **Brake**: se eliminará la potencia del motor y se colocará una carga eléctrica pasiva en el motor. Esto generalmente se hace conectando en cortocircuito los terminales del motor. Esta carga absorberá la energía de la rotación de los motores y hará que el motor se detenga más rápido que el deslizamiento.
    - **Coast**: la potencia se eliminará del motor y se detendrá libremente.
    - **Hold**: no elimina la potencia del motor. En cambio, intenta activamente mantener el motor en la posición actual. Si una fuerza externa intenta girar el motor, este retrocederá para mantener su posición. Este modo

frena la rotación del eje del motor cuando se da la orden de parar, y le indica al motor que intente mantener la posición aplicando un par al eje de giro, en sentido contrario a la fuerza externa aplicada al eje.

4.8.2.2 Sensor táctil

Es un dispositivo analógico que detecta el momento en que se presiona o se suelta el botón del sensor. Cuenta con tres estados: presionado, liberado y en contacto. Dentro de lo que ofrece EV3DEV se encuentra la interfaz "**TouchSensor**" que permite saber, a través de un valor lógico, si fue presionado o no. Esto se logra con la primitiva "**is_pressed**".

4.8.2.3 Sensor de color

Es un sensor digital que puede detectar el color o la intensidad de la luz que entra por una pequeña ventana ubicada en una de sus caras. Tiene la capacidad de distinguir hasta siete colores digitales: negro, azul, verde, amarillo, rojo, blanco y marrón. Puede utilizarse en tres modos que se mencionan seguidamente:

- **Modo color**: reconoce los siete colores antes mencionados.
- **Modo intensidad de la luz reflejada**: evalúa la fuerza de la luz que se refleja desde una lámpara emisora de luz color rojo. Sus valores oscilan entre 0 y 100, donde 0 es muy oscuro y 100 muy luminoso. El sensor debe estar en un ángulo recto y cerca de la superficie para que sus valores sean exactos. Este modo es muy utilizado para el seguimiento de línea y otros.
- **Modo intensidad de la luz ambiental**: evalúa la fuerza de la luz que ingresa en una ventana desde su entorno, como la luz del sol o el haz de una lámpara. Sus valores oscilan entre 0 y 100, donde 0 es muy oscuro y 100 muy luminoso.

En EV3DEV estos modos se utilizan haciendo uso de la interfaz que provee y sus métodos correspondientes:

Modo color	ColorSensor.MODE_COL_COLOR
Modo intensidad de la luz reflejada	ColorSensor.MODE_COL_REFLECT
Modo intensidad de la luz ambiental	ColorSensor.MODE_COL_AMBIENT

Tabla 4.4: Interfaz ev3dev para el sensor de color

A través de algoritmos establecidos se puede utilizar el sensor de color para realizar seguimientos de línea configurandolo en el modo "intensidad de la luz reflejada". Existen seis formas de concretar el seguimiento de línea del robot (Sluka J., 2009), que son las que siguen:

- **Simple o de dos niveles:** realiza un movimiento repetido de izquierda y derecha en función a la lectura del sensor, generando, de esta manera, un desplazamiento en zig zag.

- **Optimizado:** igual al anterior, solo que en esta forma se le aplica una potencia menor al motor, realizando un recorrido más suave.
- **De 5 estados:** dentro de un loop se chequea repetidamente el sensor de luz, arrojando valores de 0 a 100, siendo 0 totalmente oscuro y 100 totalmente blanco. El objetivo es llevar un valor de 50, lo que indica que el sensor se encuentra exactamente centrado en el borde de la línea. Se categoriza al sensor en cinco rangos, logrando, así, reacciones más espontáneas a la hora de corregir su dirección.
- **Proporcional:** aplica la técnica PID (Proporcional, Integral y Derivativo), la variable a medir es sensada y comparada con el valor de referencia. El controlador calcula el valor de la corrección a aplicar en base al valor del error; es decir, la acción de control se obtendrá de multiplicar el valor del error por un valor determinado denominado constante proporcional**.**
- **Proporcional e integral:** es igual al anterior solo que a este se le adiciona el algoritmo que provee el modo de control “integral” a fin de mejorar la respuesta del controlador.
- **Derivativo:** sumado al proporcional e integral se le adiciona otro algoritmo que, en este caso, provee el modo de control “derivativo”.

Según estos algoritmos, mientras más exacto es el seguimiento de línea, más lo es el procesamiento que debe realizar el robot. Por ese motivo en este proyecto se eligió el “seguimiento de línea proporcional” debido a su sencillez y también porque los lugares por donde se va a desplazar el robot son rectos y no necesita mucha lógica para su desplazamiento.

4.8.2.4 Sensor infrarrojo

Es un sensor digital que percibe la luz infrarroja reflejada por objetos sólidos, también puede detectar señales enviadas por la baliza infrarroja incluida en el kit de Lego Mindstorms. Al igual que el sensor de color, tiene tres modos de utilización:

- **Modo proximidad**: utiliza las ondas de luz reflejadas por un objeto para calcular la distancia entre el sensor y el objeto mencionado. Informa la distancia con el objeto entre valores 0 como muy cerca y 100 muy lejano. Este valor no es un número específico en centímetros o pulgadas. Detecta objetos a una distancia máxima de 50 cm.
- **Modo baliza**: solo puede utilizarse con la baliza que viene en el kit de Lego. Esta puede mandar señales hasta 200 cm, puede detectar la orientación y distancia. La orientación posee valores entre -25 y 25, siendo 0 el valor que indica que la baliza se encuentra directamente delante del sensor.
- **Modo remoto**: puede detectar que se ha presionado la baliza infrarroja remota ejecutando alguna acción al recibir esta señal.

Al igual que el sensor de color, EV3DEV provee una interfaz para hacer uso de estos modos en el robot:

Modo proximidad	InfraredSensor.MODE_IR_PROX

Modo baliza	InfraredSensor.MODE_IR_CAL
Modo remoto	InfraredSensor.MODE_IR_REMOTE

Tabla 4.5: Interfaz ev3dev para el sensor infrarrojo

A diferencia del sensor ultrasónico, posee una menor distancia de recepción de objetos sólidos, aunque para este proyecto no se necesitaba mucha distancia.

4.8.3 Configuración de la ciudad física

La configuración de la ciudad física se puede realizar desde la interfaz GUI suministrada por el entorno de desarrollo EDIS, o bien, manipulando, de forma manual, el archivo de configuración establecido por el intérprete DaVinci-C.

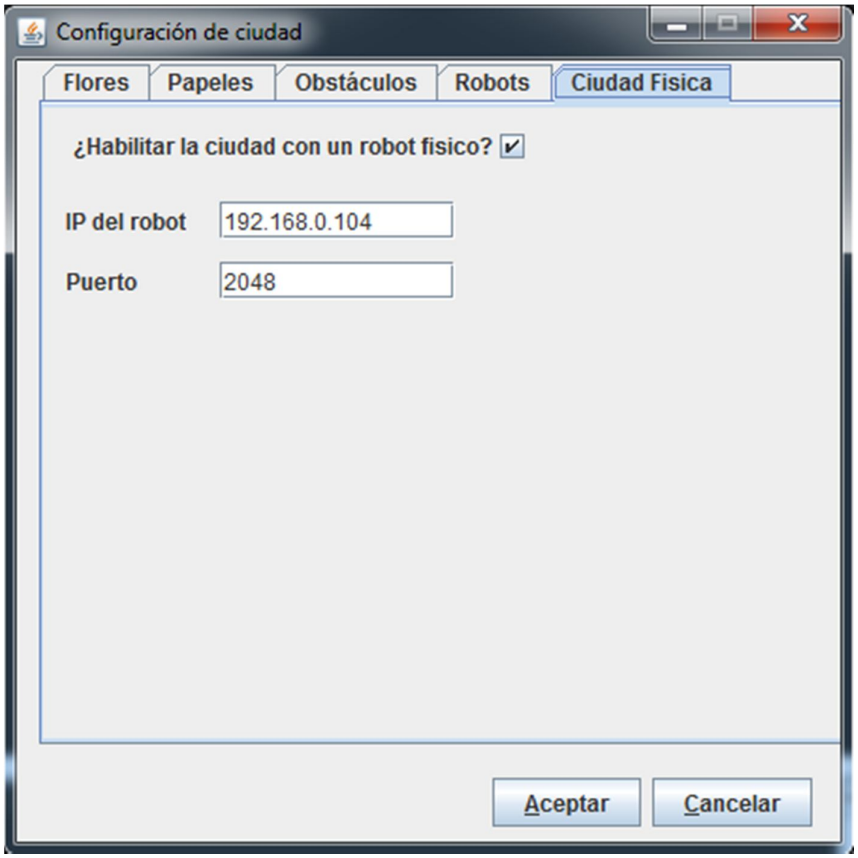


Figura 4.11: Configuración de la ciudad física a través de la interfaz gráfica

En la nueva pestaña se pueden ver los siguientes agregados:

- "Habilitar ciudad", prepara los métodos necesarios para abrir una comunicación con el robot, según el protocolo establecido

- “IP del robot”, dirección única en la red compartida donde se encuentra el robot conectado a través de wifi.
- “Puerto”, puerto de comunicación para conectarse con el robot.

4.8.4 Configuración del sistema operativo EV3DEV

El sistema operativo EV3DEV es una distribución de Linux basada en Debian, actualmente se encuentra su versión 8.5, y está adaptada para ejecutarse en varias plataformas de Lego Mindstorms y Raspberry Pi. Su instalación se realiza en una tarjeta micro SD que luego es insertada en las placas y/o bloques programables antes mencionados. Las instrucciones para su instalación y uso en el bloque programable Lego Mindstorms EV3 se detallan a continuación:

1. Descargar el sistema operativo EV3DEV
    a. Descargar imagen de su sitio oficial (http://www.ev3dev.org)
2. Grabarlo en una tarjeta microSD
    a. Descargar e instalar el programa **Etcher** que ayudará a grabar la imagen en la tarjeta micro SD.
    b. Ejecutar Etcher después de que haya terminado de instalarse.
    c. Seleccionar la imagen del sistema operativo EV3DEV descargado previamente.
    d. Conectar la tarjeta SD a su PC, luego seleccionar como unidad de destino para su instalación.
    e. Una vez comprobado todos pasos anteriores haga clic en “grabar” para completar la instalación.
3. Insertar la tarjeta micro SD, con el sistema operativo en ella, en el bloque programable EV3 y luego encenderla.
4. El bloque programable EV3 arrancará normalmente y luego de unos minutos cargará el nuevo sistema operativo instalado.
    a. Es normal que el primer arranque dure mucho más que lo habitual.
5. El nuevo sistema operativo ya estará cargado en su micro SD y los futuros arranques del bloque programable serán con EV3DEV.

4.8.4.1 Conexión vía SSH

El bloque programable Lego Mindstorms EV3 tiene un puerto host USB, por ende, se le puede insertar un adaptador wifi USB para conectarlo a una red compartida e ingresar por el protocolo SSH. Al conectarlo, el sistema EV3DEV lo reconocerá automáticamente y le brindará una interfaz para conectarlo a una red dándole su IP correspondiente.

protocolo: SSH
puerto: 22
usuario: robot
clave: maker

4.8.4.2 Instalación de la librería ev3dev-lang para Python3

La librería ev3dev-lang para Python es una interfaz para dispositivos EV3DEV que permiten controlar motores, sensores, pantallas LCD y más, desde el código Python. Para poder hacer uso de ella se debe descargar el paquete desde su página oficial e instalar en el bloque programable. Los pasos son los siguientes:

1. Ingresar por ssh al dispositivo
2. Descargar la librería

```
robot@ev3dev:~$ wget https://github.com/ev3dev/ev3dev-lang-python/archive/jessie.zip
```

3. Descomprimir e ingresar a la carpeta

```
robot@ev3dev:~$ gzip -d jessie.zip
robot@ev3dev:~$ cd jessie/
```

4. Instalar

```
robot@ev3dev:~$ python setup.py install
```

4.8.5 Componentes utilizados

Para el ensamblaje del robot final se utilizaron un total de noventa y ocho componentes incluidos en el kit, a esto se le sumó una tarjeta micro SD de 4 GB y un adaptador USB wifi para brindar la conexión correspondiente.

En el Anexo II Componentes utilizados se detalla cada uno de los componentes que fueron utilizados para el ensamble del robot.

4.8.6 Ensamble general

4.8.6.1 Ensamble del prototipo

El nombre del robot quedó establecido como DVC-Lin3. Para la construcción de su prototipo se utilizó la aplicación llamada Lego Digital Designer, también conocida como LDD. Se trata de una herramienta que permite diseñar cualquier robot Lego desde cero con una cantidad ilimitada de piezas para su construcción.

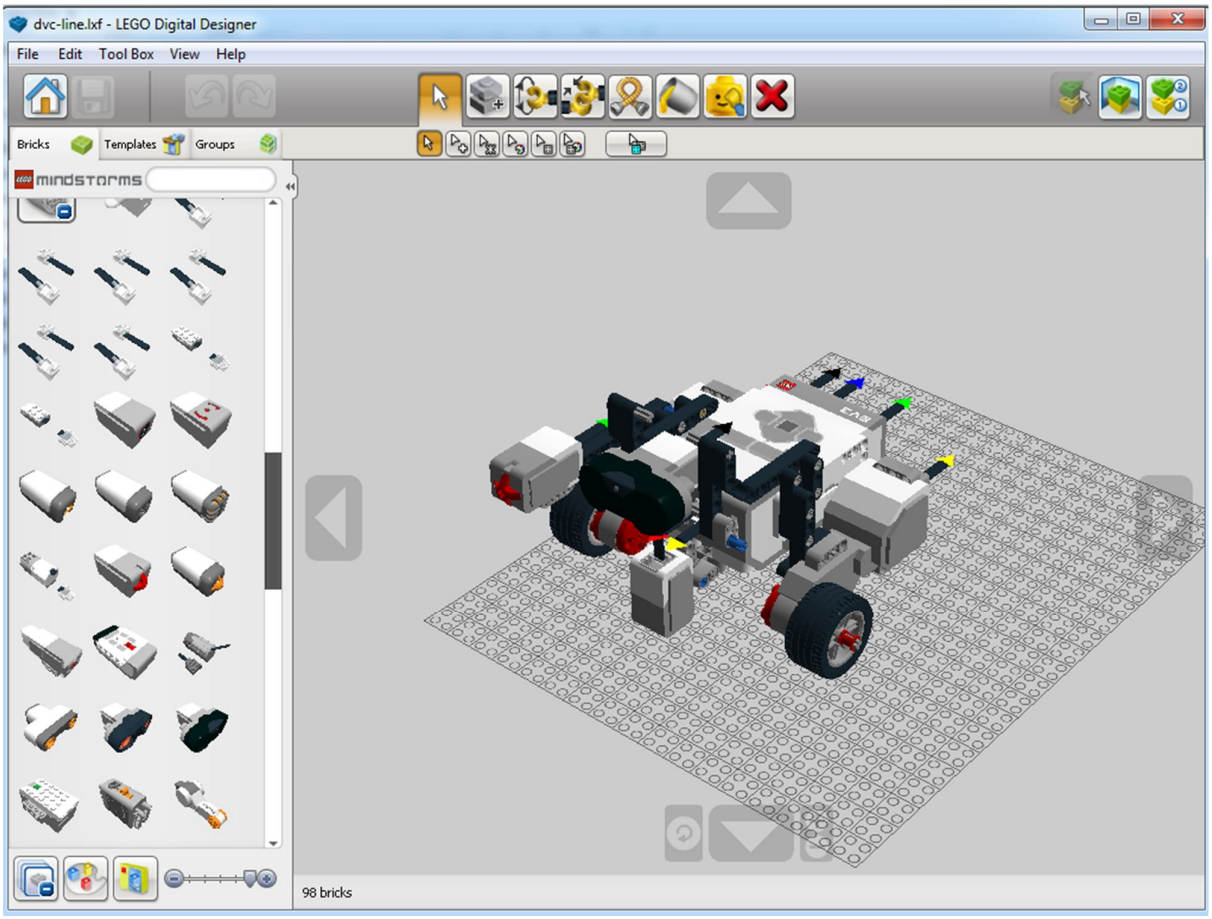


Figura 4.12: Lego Digital Designer - Ensamble del prototipo

En el Anexo III Ensamble del robot DVC-Lin3, se detalla paso a paso su armado.

4.8.6.2 Vistas finales

A continuación se muestran diferentes perspectivas del robot Lego Mindstorms EV3 ensamblado listo para cumplir su programación final.

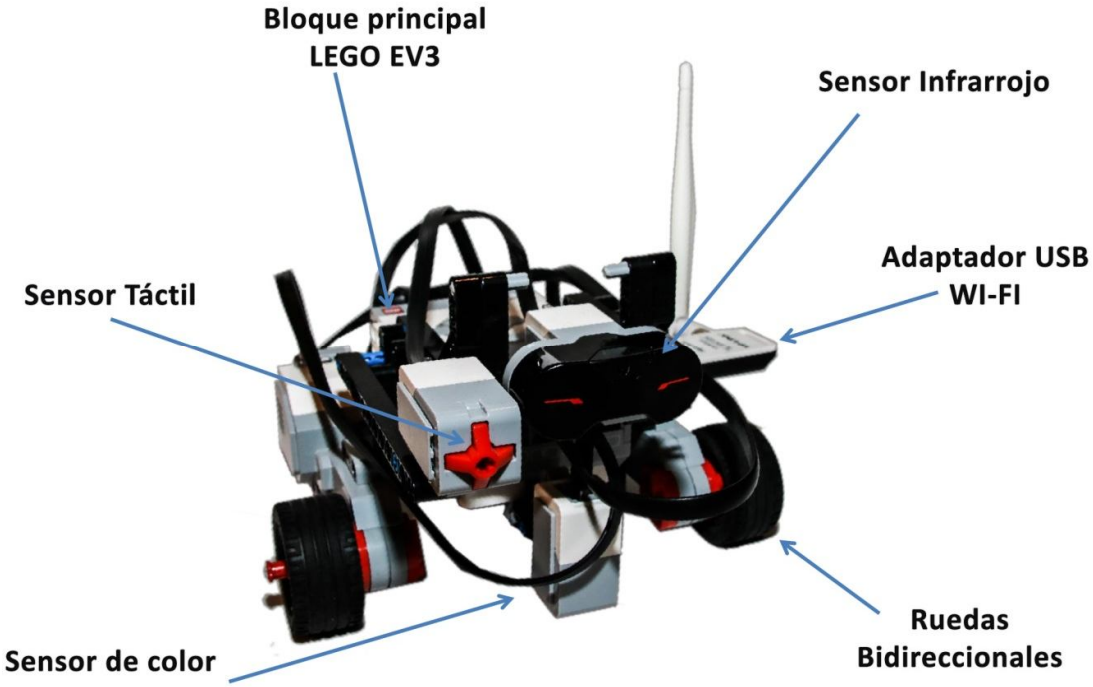


Figura 4.13: Vista perspectiva DVC Lin3

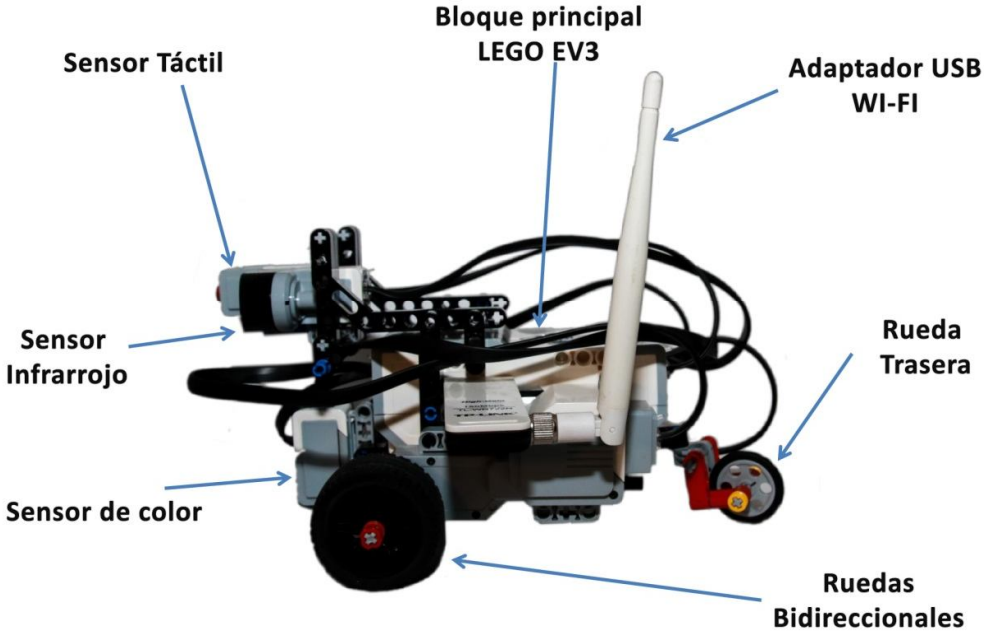


Figura 4.14: Vista lateral izquierda DVC Lin3

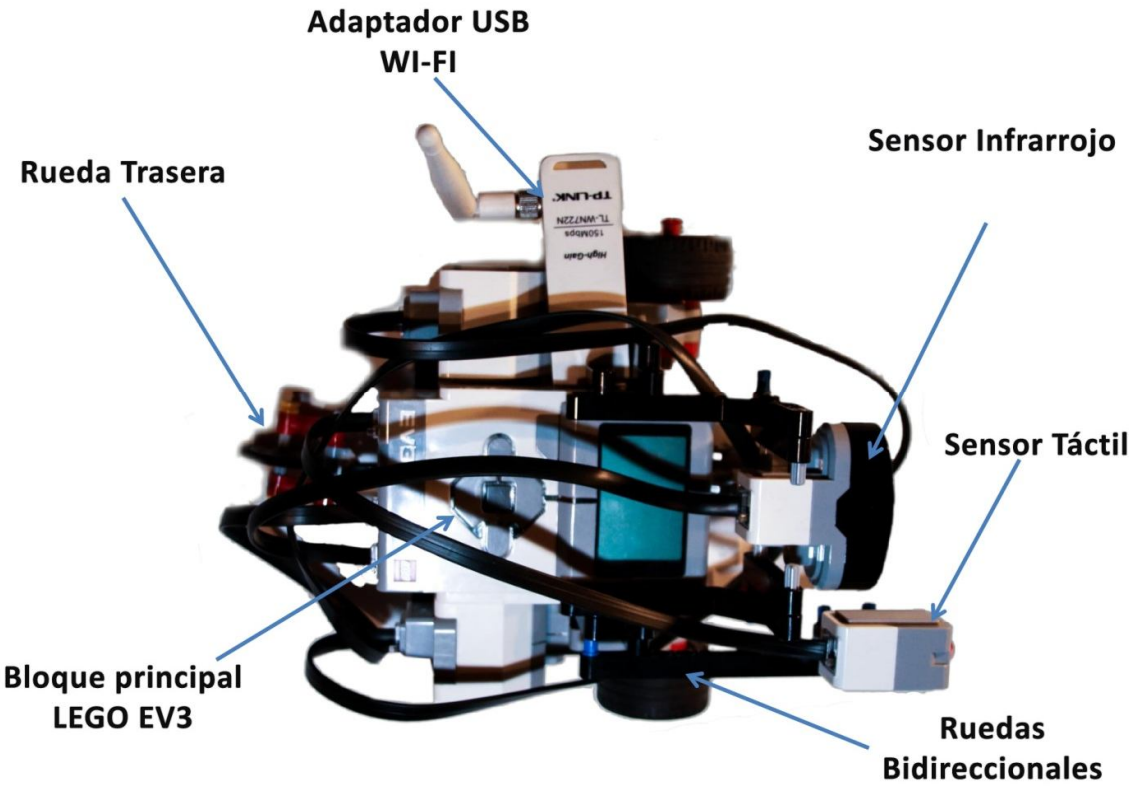
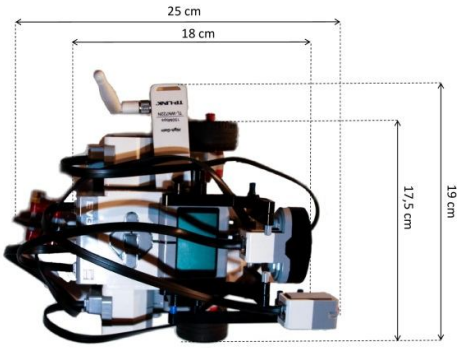
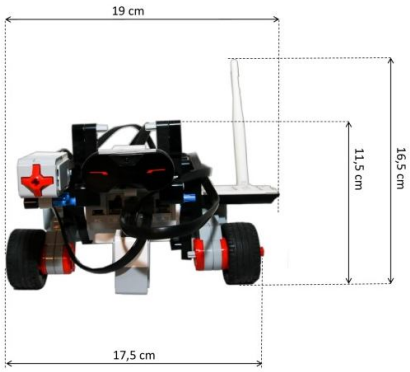


Figura 4.15: Vista superior DVC Lin3

4.8.7 Esquema y dimensiones



Vista superior



Vista frontal

4.9 Ejemplos aplicables al robot Lego EV3 - DVC-lin3

4.9.1 Estructuras de selección simple o condicional

En el siguiente programa el robot avanza una cuadra, **SI** existe un obstáculo en la próxima esquina, gira a la derecha y avanza una cuadra más, **SINO** continúa su recorrido por una cuadra más

```
programa obstaculo

comenzar
     iniciar
     mover
     si (hayObstaculo)
          comenzar
               informar("se encontró un obstáculo")
               derecha
               mover
          fin
     sino
          comenzar
               informar("no hay obstáculos")
               mover
          fin
fin
```

4.9.2 Estructuras de iteración

4.9.2.1 Repetitiva

En el siguiente programa el robot realiza un cuadrado de 2 x 2 terminando en su posición inicial.

```
programa cuadrado

comenzar
     iniciar
     repetir 4
          comenzar
               repetir 2
                    mover
               derecha
          fin
fin
```

4.9.2.2 Condicional

En este programa el robot realiza su recorrido en un área cuadrada, y **MIENTRAS** no aparezca un obstáculo en cualquiera de sus esquinas, el robot continuará su movimiento.

```
programa pruebaMientras

comenzar
      iniciar
      mientras !hayobstaculo
            comenzar
                  mover
                  derecha
            fin
fin
```

4.9.3 Subprogramas. Procedimientos

En el programa que sigue el robot utiliza subprogramas, realiza una escalera con la cantidad de escalones que es ingresada por el usuario mediante una variable de tipo número llamada “escalones”

```
programa escalera

subprogramas

procedimiento izquierda
comenzar
      repetir 3
            derecha
fin

variables
  escalones: numero

comenzar
      iniciar
      pedir(escalones)
      repetir escalones
      comenzar
            mover
            derecha
            mover
            izquierda
      fin
fin
```

Cabe destacar que en los ejemplos descritos anteriormente se puede visualizar que el robot es capaz de ejecutar cualquier tipo de programa que contenga dentro de su código estructuras de control, subprogramas (procedimientos y/o funciones), funciones primitivas del lenguaje y cualquier declaración léxica que el estudiante pueda incluir en el algoritmo que esté desarrollando.

5. Conclusiones

Vivimos en una era donde la sociedad y la tecnología crecen a un ritmo acelerado, por lo que es necesario adaptarse a estos cambios. La enseñanza de la programación en personas de temprana edad es un factor clave para integrarse a estos nuevos modelos. Es una realidad mencionar que el futuro del sistema laboral tiende a estar ligado a los sistemas tecnológicos. Es notable la falta de programadores en el ámbito laboral como así también su creciente demanda. La programación es una herramienta fundamental en el crecimiento de la sociedad, por este motivo fomentar este afán educacional es algo que apasiona.

Mejorar un intérprete ya desarrollado implicó un gran trabajo de investigación para entender su funcionamiento y visualizar las mejoras que se le podrían realizar, siempre poniendo foco en el fortalecimiento del pensamiento computacional del estudiante. Particularmente, el uso de funciones le dará las herramientas para incorporar algunos conceptos básicos en la programación estructurada, que le servirá para futuros paradigmas y lenguajes que decida aprender.

Desarrollar y fomentar cuestiones vinculadas a la programación abrió un camino para visualizar nuevas estrategias que permitirán estimular la enseñanza en los estudiantes. A través de la robótica educativa se incorporan nuevas vías de aprendizaje para la educación, con más recursos y nuevos métodos de enseñanza.

El robot educativo programable seleccionado para esta tesis, pertenece a la empresa Lego Mindstorms, contiene las características y recursos necesarios para el objetivo que se buscaba. El único punto en contra es su elevado costo al compararlo con otros robots educativos de similares o iguales características. La facilidad de su diseño y construcción lo presentan como el ideal en un ámbito donde la electrónica no es el punto fuerte, sino que se utiliza como una capa para la enseñanza de la programación.

El diseño y armado del robot también fue todo un desafío. Si bien en internet existen muchas guías de modelos terminados, se lo debía adaptar exclusivamente a las necesidades que el proyecto ameritaba: la ubicación de las ruedas principales, la posición de los sensores y la estructura en general. En este punto la empresa Lego también fue muy visionaria en desarrollar la herramienta, llamada LEGO Digital Designer, que facilitara el diseño previo al armado.

El estudio y aprendizaje de las herramientas necesarias para realizar el intérprete en el robot demandó gran parte del tiempo de todo el proyecto, debido a que, si bien existía mucha documentación, se trataba de un lenguaje de programación totalmente nuevo. Aun así, con las bases de programación aprendidas en la Universidad, durante toda la carrera, esta tarea se volvió mucho más admisible y llevadera, logrando de esta manera cumplir con el objetivo planteado.

5.1 Trabajos futuros

A medida que el proyecto fue avanzando surgieron cuestiones para planificar y desarrollar en futuras etapas. Si bien se logró definir e implementar con éxito las funciones dentro del lenguaje de programación DaVinci-C, quedó pendiente la implementación de funciones recursivas para la contemplación de mayores casos de desarrollos. Otra posible implementación para el futuro es realizar nuevos intérpretes en otras plataformas de robots, por ejemplo, Arduino o Raspberry Pi, para que junto con el robot actual, el estudiante pueda realizar algoritmos más complejos que se reflejen en la ciudad física, lo que potenciaría su pensamiento computacional.

Otro punto importante para mejorar más adelante es el inconveniente que surgió a partir del desgaste de la batería en los motores puesto que, al no contar con un segundo sensor de color, la finalización del recorrido en las primitivas que hacían uso de los motores del robot, se determinaban con una variable interna que indicaba su posicionamiento. Entonces, al desgastarse la batería el rendimiento de los motores decaía, por ende, en algoritmos grandes los resultados finales no eran los esperados debido a que el robot no lograba completar su recorrido.

También habría que considerar otras posibilidades de desarrollo, no menos significativas, como las que se detallan a continuación:

- Agregar un sensor de color en la parte frontal del robot para que reconozca fehacientemente el fin de cada cuadra.
- Analizar la inclusión de un segundo robot en la ciudad para iniciar a los alumnos en problemas de concurrencia
- Con más recursos de hardware, establecer primitivas que permitan la recolección de flores y papeles, integrando una ciudad inteligente y/o pensando otros mecanismos.
- Optimizar la primitiva MOVER, actualmente el seguimiento de línea lo hace en un solo lado, si se lograra modificar el intérprete para que reconozca cualquier lado, disminuiría la cantidad de errores y pérdidas de línea del robot.

Es de destacar que parte de los futuros desarrollos citados en la tesis, son considerados para ser implementados dentro del proyecto *Diseño y Aplicación de Estrategias para la Enseñanza Inicial de la Programación*, del cual forma parte de la unidad ejecutora.

6. Herramientas utilizadas

El conjunto de herramientas utilizadas para el desarrollo de esta tesis, se detallan a continuación. (Cada una forma parte de la construcción general del mismo).

6.1 Codificación y programación

Netbeans, es un entorno de desarrollo utilizado para la modificación del intérprete DaVinci-C al igual que el entorno de desarrollo llamado EDIS.

Sublime Text, se trata de un editor de texto muy completo que incluye realce de sintaxis para múltiples lenguajes de programación, el cual se utilizó para la programación del intérprete del robot en el lenguaje de programación Python.

6.2 Documentación y almacenamiento

Dropbox, refiere a un servicio de alojamiento en la nube, posee un sistema de versionado de archivo que se utilizó para el resguardo de todos los documentos vinculados al proyecto.

Google Docs, es un procesador de texto online perteneciente a la empresa Google, posee herramientas que permiten el trabajo colaborativo muy efectivo. Se utilizó para la creación del documento principal y sus correcciones.

Microsoft Word, procesador de texto con el cual se brindó la estética final del documento permitiendo configurar las normas de citas, márgenes de impresión, índices, entre otras características.

Microsoft PowerPoint, se trata de una aplicación de presentaciones utilizada para realizar la presentación final del proyecto.

Sourceforge, sitio web de colaboración para proyectos de software que fue utilizado para resguardar el proyecto final, crear una página informativa y brindar la posibilidad a cualquier programador que lo desee a contribuir con él.

6.3 Edición de multimedia

yEd Graph Editor, herramienta para la creación de gráficos utilizada para la generación de los gráficos provistos en el documento final.

Adobe Lightroom, es una aplicación para mejorar imágenes con la que resaltaron los colores de las fotos del robot en su versión final.

Adobe Photoshop, editor de gráficos que permitió quitar los fondos de las imágenes del robot y generar una mejor impresión.

Camtasia Studio, editor de video con el que se prepararon las demostraciones del prototipo funcionando en su versión final.

6.4 Prototipos

LEGO Digital Designer, es un software libre desarrollado por la empresa The Lego Group, aplicado en la creación del diseño preliminar del robot y la inclusión, paso a paso, de su armado.

Autodesk Autocad, software de diseño asistido por computadora utilizado para dibujo 2D y modelado 3D. Se empleó para diseñar e imprimir el diseño de la ciudad final donde se desplazaría el robot.

6.5 Conectividad

Putty, es un cliente que acepta diferentes protocolos de comunicación, entre ellos SSH, con el cual se pudo lograr la comunicación con el robot para realizar la implementación del intérprete.

WinSCP, se trata de un cliente SFTP gráfico para Windows que emplea SSH, se lo aprovechó para hacer una copia, cada cierto período, de los archivos del robot hacia la computadora central del proyecto.

7. Referencias

Aguil Mallea D. E., «Un Intérprete Multiplataforma Para La Iniciación A La Programación Estructurada Y Concurrente», Tesis de grado, Ushuaia, 2012. [En línea]. Disponible en http://davinci-c.sourceforge.net/ [Último acceso: junio 2018]

Baturone A. O., Robótica: manipuladores y robots móviles, Marcombo, cap. 1, pag. 1-3, 2001.

Barber Castaño R. I., «Desarrollo de una arquitectura para robots móviles autónomos: aplicación a un sistema de navegación topológica», cap 2, pag. 9-11, Leganés, España, 2000.

Berlanga Llavori R. y Quereda J. M. I., Introducción a la programación con Pascal, Universitat Jaume, cap. 5, pag. 109-112, 2000.

Champredonde R., «Visual Da Vinci, Lenguaje y Ambiente Visual para la Enseñanza de Programación en el Curso de Ingreso y en el Primer Año de las Carreras de Informática» Tesis de grado, La Plata, 1997. [En línea]. Disponible en http://sedici.unlp.edu.ar/bitstream/handle/10915/2170/Documento_completo__.pdf?sequence=1 [Último acceso: junio 2018]

Cuero Sandoval E., «Introducción A La Robótica Educativa,», 2015 [En línea]. Disponible en https://es.calameo.com/read/002448031c4ab8d1e2de9 [Último acceso: junio 2018].

Englebardt S. L., «Men to feed computers» Popular Science, EEUU, pag. 106-109, 1965.

Hempel R., Lechner D., «EV3DEV» 2015. [En línea]. Disponible en: http://www.ev3dev.org [Último acceso: diciembre de 2017].

Knasel T. M., «Mobile robotics—State of the art review.,» Elsevier Science Publishers, vol. 2, pag. 149-155, 1986.

Osorio J. A. C., Salazar O. A. y Mendoza J. A., «El entorno LEGO Mindstorms en la introducción a la robótica y la programación» de Scientia et Technica Año XV, pag. 42-45, 2009.

Rodríguez Sala J. J., «Introducción a la programación. Teoría y práctica», pag. 4, España, 2003

Ruiz Velasco E. S., La robótica pedagógica. En Educación, universidad y sociedad: el vínculo crítico, pag. 117-148, Barcelona, 2004.

Sluka, J. «A PID Controller For LEGO MINDSTORMS Robots» 2009. [En línea]. Disponible en: http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html. [Último acceso: enero de 2018].

Tejero C. G., «Un Entorno De Desarrollo Integrado Multiplataforma Para La Iniciación A La Programación Estructurada Y Concurrente», Tesis de grado, Ushuaia, 2013.

7.1 Bibliografía

Arjona Fernández M. L., Revista educación digital, «Importancia y elementos de la programación didáctica» Hekademos, vol. III, nº 7, Diciembre 2010.

Arkin R. C., Behavior-based robotics, MIT press, 1998.

Artero Ó. T., Arduino: curso práctico de formación, RC Libros, 2013.

Calvo I. y Gorka P., «Uso conjunto de la plataforma LEGO Mindstorms NXT y metodologías PBL en informática industrial.,» 2010.

Champredonde R. y De Giusti A., «Design and Implementation of Visual Da Vinci» de III Congreso Argentino en Ciencias de la Computación, pag. 2-3, La Plata, 1997.

De Pablos J. A. y Pina Calafi A., «Enseñanza-aprendizaje constructivista a través de la Robótica Educativa,» de CiDd: II Congres Internacional de Didactiques, Navarra, 2010.

Depetris B. O., Aguil Mallea D., Pendenti H., Tejero G. y Feierherd G., «Experiencias con Da Vinci Concurrente en la enseñanza inicial de la programación secuencial y la programación concurrente,» de VII Congreso – Tecnología en educación y educación en tecnología, 2013.

Depetris B. O., Aguil Mallea D., Pendenti H., Tejero G., Feierherd G., Prisching G., Fierro A., Aguilar S., Domínguez J. y Mamaní J., «Diseño y Aplicación de Estrategias para la Enseñanza Inicial de la Programación», Workshop de Investigadores en Ciencias de la Computación, 2018

Goodrich M. A., Experiments in adjustable autonomy., Seattle: MIT Press, 2001.

Joyanes Aguilar L., Fundamentos de programación: algoritmos y estructura de datos y objetos, 2003.

Kelleher C. y Pausch R., Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers, 2003.

Lacoa R. F., Lacoa J. F. y Blair A., «La enseñanza de Lenguajes de Programación en la Escuela,» Mayo 2016.

Lamoyi Villamil L., «La robótica Lego Mindstorms®: un recurso didáctico para fortalecer el pensamiento lógico matemático,» 2014.

Leal T., Edgar, Guerrero Melendez T. Y. y Saldivar Alonso V. P., «Revisión de la plataforma robótica LEGO Mindstorms para aplicaciones educativas y de investigación,» de Sistemas & Telemática, pag. 9-27, 2013.

Melamed A., «El futuro del trabajo y el trabajo del futuro,» Planeta Argentina, 2017.

Muñoz R., «Uso de Scratch y Lego Mindstorms como apoyo a la docencia en Fundamentos de programación,» de Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática, La Salle, 2015.

Papert S., Mindstorms: Children, computers, and powerful ideas, 1980.

Pfeifer R. y Scheier C., Understanding intelligence, MIT press, 2001.

Pinto Salamanca M. L. y Giovanni R. B. B., «Determinación de parámetros de un robot móvil de lego mindstorms®,» de Ingeniería Investigación y Desarrollo, Sogamoso, Colombia, 2007, pag. 7-12.

Resnick E. A., Scratch: Programming for all, 2009.

Richardson M. y Shawn W., Getting started with raspberry PI, 2012.

Romero Costas M., Robótica : Entrá al mundo de la inteligencia artificial, Buenos Aires: Educ.ar S.E., 2012.

Sánchez F. Á. B. y Guzmán A. F., La robótica como un recurso para facilitar el aprendizaje y desarrollo de competencias generales, 2012.

Toledo P. F., Robótica con Lego Mindstorms, 2009.

7.2 Tesis consultadas

Gómez Ortiz I., «Librería de comunicaciones para control remoto de robots,» 2017.

Ortiz Martínez D., «Robótica para seguimiento de líneas,» Cataluña, 2016.

Vázquez de la Presilla Á., «Aplicación cliente servidor para el control del robot,» 2017.

7.3 Páginas web

«Arduino,» [En línea]. Disponible en: https://www.arduino.cc/en/Guide/Environment. [Último acceso: Diciembre 2017].

«Área Tecnológica,» [En línea]. Disponible en: http://www.areatecnologia.com/electronica/tipos-de-robots.html. [Último acceso: Diciembre 2017].

«Clasificacion de robots | Wiki de Robótica - UPV.» [En línea]. Disponible en: http://wiki.robotica.webs.upv.es/wiki-de-robotica/introduccion/clasificacion-de-robots/. [Último acceso: Noviembre 2017].

«Comohacer,» [En línea]. Disponible en: https://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/. [Último acceso: Enero 2018].

«Edukative,» [En línea]. Disponible en: https://edukative.es/que-es-la-robotica-educativa/. [Último acceso: Diciembre 2017].

«EsMindstorms,» [En línea]. Disponible en: http://www.esmindstorms.com/. [Último acceso: Enero 2018].

«Lego Mindstorms,» [En línea]. Disponible en: https://www.lego.com/en-us/mindstorms. [Último acceso: Enero 2018].

«Lignux,» [En línea]. Disponible en: https://lignux.com/lenguaje-maquina/. [Último acceso: 2017 Julio].

«Paradigma,» [En línea]. Disponible en: https://www.paradigmadigital.com/dev/la-programacion-funcional-deberias-usarla/. [Último acceso: Diciembre 2017].

«Snoop Consulting» [En línea]. Disponible en: http://www.snoopconsulting.com/. [Último acceso: Diciembre 2017].

«Universidad Internacional de Valencia,» 2018. [En línea]. Disponible en: https://www.universidadviu.es/lenguaje-nivel-caracteristicas-funciones/. [Último acceso: Junio 2017].

T. Renzo y N. Imideo, «Educomuicacion,» [En línea]. Disponible en: http://www.educomunicacion.es/didactica/0031clasificacionmetodos.htm. [Último acceso: Agosto 2017].

Anexos

I - Código fuente interprete robot

A continuación se mencionan porciones de código correspondientes al intérprete que reconoce y ejecuta las instrucciones descritas en el robot físico.

Instrucción: iniciar
Código Fuente:

```
    def __init__(self):
        self.motord = None
        self.motori = None
        self.motores = []
        self.sensorc = None
        self.sensori = None
        self.sensort = None
        self.sonido = None

    # reset de motores
    def __reset_motores(self):
        for motor in self.motores:
            motor.reset()
            motor.position = 0
            motor.speed = 500

    # inicializacion de motores
    def __iniciar_motores(self):
        self.motord = ev3.LargeMotor(ev3.OUTPUT_B)
        self.motori = ev3.LargeMotor(ev3.OUTPUT_C)
        self.motores = [self.motord, self.motori]
        assert all([motor.connected for motor in self.motores]),
"Motores no conectados!!"
        self.__reset_motores()

    # inicializacion de sensor de color
    def __iniciar_sensor_color(self):
        self.sensorc = ev3.ColorSensor()
        self.sensorc.mode = ev3.ColorSensor.MODE_COL_REFLECT
        assert self.sensorc.connected, 'Sensor de color no
detectado!!'

    # inicializacion de sensor infrarojo
    def __iniciar_sensor_infrarojo(self):
        self.sensori = ev3.InfraredSensor()
        self.sensori.mode = ev3.InfraredSensor.MODE_IR_PROX
        assert self.sensori.connected, 'Sensor de infrarojo no
detectado!!'

    # inicializacion de sensor tactil
    def __iniciar_sensor_tactil(self):
        tactil = ev3.TouchSensor()
        assert tactil.connected, 'Sensor tactil no detectado!!'

    # inicializacion de sonido
    def __iniciar_sonidos(self):
        self.sonido = ev3.Sound
```

```
    # iniciar
    def iniciar(self):
        self.__iniciar_motores()
        self.__iniciar_sensor_color()
        self.__iniciar_sensor_infrarojo()
        self.__iniciar_sensor_tactil()
        self.__iniciar_sonidos()

        self.sonido.play('sounds/iniciar.wav').wait()
        self.sonido.beep()

        return respuestas.OK
```

Instrucción: mover

Código Fuente:

```
def mover(self):
  self.__reset_motores()

  #Constantes
  kp = 0.5    #Constante de proporcionalidad

  minv = 10   #Punto minimo del rango de valores (blanco es 0 )
  maxv = 80   #Punto maximo del rango de valores (Negro es 100)
  esperado = (maxv + minv) / 2   #Punto medio
  potencia  = 50  #Potencia objetivo de los motores

  self.motori.run_direct(duty_cycle_sp=50)
  self.motord.run_direct(duty_cycle_sp=50)

  #Bucle principal, se detiene al presionar el boton tactil
  while (not(self.sensort.value()) and (self.motori.position < 430 and
self.motord.position < 430)):
    luz         = self.sensorc.value() #Leemos el sensor de luz
    error       = luz - esperado       #Calcula el error
    correccion  = kp * error
    correcciond = potencia - correccion
    correccioni = potencia + correccion

    self.motord.run_direct(duty_cycle_sp=correcciond)
    self.motori.run_direct(duty_cycle_sp=correccioni)

  self.motord.stop(stop_action="brake")
  self.motori.stop(stop_action="brake")
  self.__detener_motores()

  if not(self.sensort.value()):
     salida = respuestas.OK
  else:
     salida = respuestas.TTPRES
      self.sonido.play('sounds/boton-en-contacto.wav').wait()



  return salida
```

Instrucción: derecha
Código fuente:

```
def derecha(self):

  self.sonido.play('sounds/derecha.wav').wait()

  self.__reset_motores()

  if not(self.sensort.value()):

    #ADELANTAMOS AL ROBOT PARA CORREGIR LA POSICION
    self.motori.run_to_rel_pos(position_sp=90, speed_sp=150,
stop_action="hold")
    self.motord.run_to_rel_pos(position_sp=90, speed_sp=150,
stop_action="hold")
    self.motori.wait_while('running')
    self.motord.wait_while('running')

    #GIRAMOS AL ROBOT
    self.motori.run_to_rel_pos(position_sp=310, speed_sp=300,
stop_action="hold")
    self.motord.run_to_rel_pos(position_sp=-310, speed_sp=300,
stop_action="hold")
    self.motori.wait_while('running')
    self.motord.wait_while('running')


    #ATRASAMOS AL ROBOT PARA CORREGIR LA POSICION
    self.motori.run_to_rel_pos(position_sp=-90, speed_sp=150,
stop_action="hold")
    self.motord.run_to_rel_pos(position_sp=-90, speed_sp=150,
stop_action="hold")
    self.motori.wait_while('running')
    self.motord.wait_while('running')

    salida = respuestas.OK

  else:
    salida = respuestas.TTPRES
    self.sonido.play('sounds/boton-en-contacto.wav').wait()

  self.__detener_motores()


  return salida
```

Instrucción: hayObstaculo
Código fuente:

```
def hayObstaculo(self):
```

```
  if not(self.sensort.value()):
    salida = (respuestas.VERDADERO if (self.sensori.value() <= 70)
else respuestas.FALSO)
  else:
    salida = respuestas.TTPRES
    self.sonido.play('sounds/boton-en-contacto.wav').wait()

  if (salida == respuestas.VERDADERO):
     self.sonido.play('sounds/hay-un-obstaculo.wav').wait()
     self.sonido.tone([(500, 200, 400)] * 2).wait()


  return salida
```

Instrucción: terminar

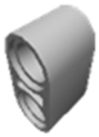
Código fuente:

```
def __detener_motores(self):
  for motor in self.motores:
    motor.stop()

def terminar(self):
  self.__detener_motores()

  self.sonido.play('sounds/conexion-finalizada.wav').wait()
  self.sonido.tone([(500, 100, 400)] * 3).wait()

  return respuestas.OK
```

II - Componentes utilizados

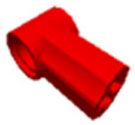
A continuación se detallan cada uno de los componentes que fueron utilizados para el ensamble del robot DVC-Lin3.

Nombre	Imagen	ID.	Código de color	Cant.
WEDGE-BELT WHEEL Ø24		4185	194 - Medium Stone Grey	1
TYRE FOR WEDGE-BELT WHEEL		2815	26 - Black	1
RIM WIDE W.CROSS 30x20		56145	194 - Medium Stone Grey	2
TYRE NORMAL WIDE Ø43,2 X 22		44309	26 - Black	2
CABLE 250 MM		11145	40 - Transparent, 26 - Black	2
CABLE 350 MM		11146	40 - Transparent, 26 - Black	2

CABEL 500 MM		11147	40 - Transparent, 26 - Black	1
MS, EV3, SENSOR, COLOUR		95650	194 - Medium Stone Grey, 1 - White, 26 - Black, 102 - Medium Blue, 21 - Bright Red	1
MS, EV3, TOUCH SENSOR		95648	194 - Medium Stone Grey, 1 - White, 315 - Silver Metallic, 21 - Bright Red	1
MS-EV3, IR SENSOR		95654	194 - Medium Stone Grey, 1 - White, 109 - Pc Black Ir, 26 - Black, 315 - Silver Metallic	1
MS 2013 ENGINE		95658	21 - Bright Red, 194 - Medium Stone Grey, 1 - White	2
MS-EV3, P-BRICK		95646	1 - White, 194 - Medium Stone Grey, 40 - Transparent, 199 - Dark Stone Grey, 315 - Silver Metallic	1
TECHNIC 2M BEAM		43857	194 - Medium Stone Grey	5
TECHNIC 3M BEAM		32523	26 - Black	2

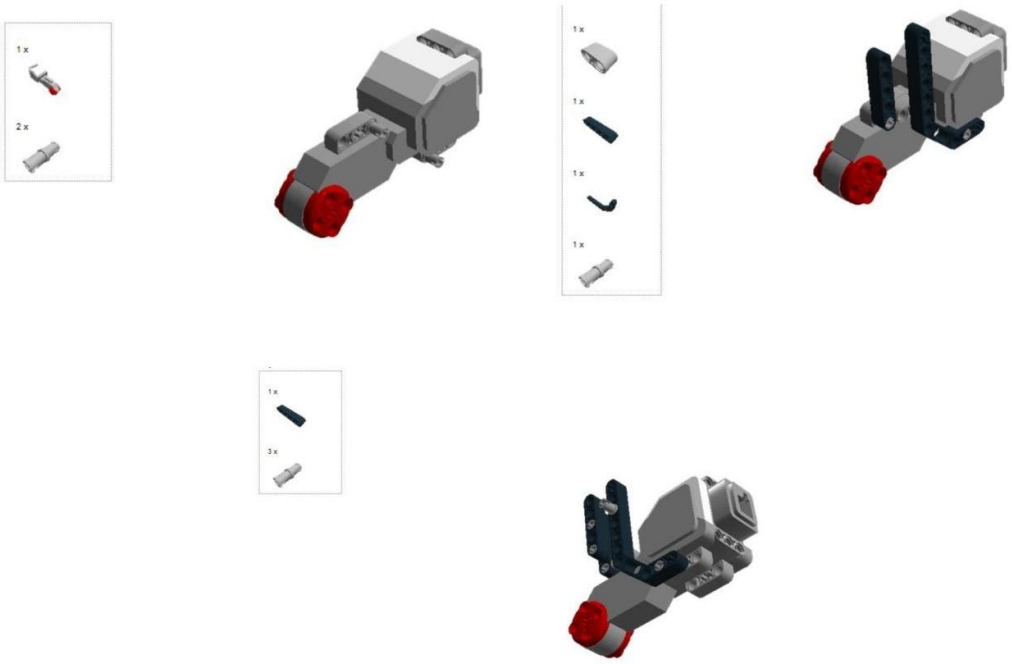
TECHNIC 5M BEAM		32316	26 - Black	5
TECHNIC 7M BEAM		32524	26 - Black	2
T-BEAM 3X3 W/HOLE Ø4.8		60484	26 - Black	1
TECHNIC 13M BEAM		41239	26 - Black	1
TECHNIC 15M BEAM		32278	26 - Black	1
TECHNIC ANG. BEAM 3X5 90 DEG.		32526	26 - Black	1
DOUBLE ANGULAR BEAM 3X7 45°		32009	26 - Black	3
1/2 BUSH		32123	24 - Bright Yellow	4

2M CROSS AXLE W. GROOVE		32062	21 - Bright Red	1
CONNECTOR PEG		3673	194 - Medium Stone Grey	30
CROSS AXLE 3M		4519	194 - Medium Stone Grey	3
CONNECTOR PEG/CROSS AXLE		6562	5 - Brick Yellow	4
BUSH FOR CROSS AXLE		6590	21 - Bright Red	2
CONN.BUSH W.FRIC./CROSSALE		43093	23 - Bright Blue	1
CONNECTOR PEG W. FRICTION 3M		6558	23 - Bright Blue	4
CROSS AXLE 5M		32073	194 - Medium Stone Grey	4

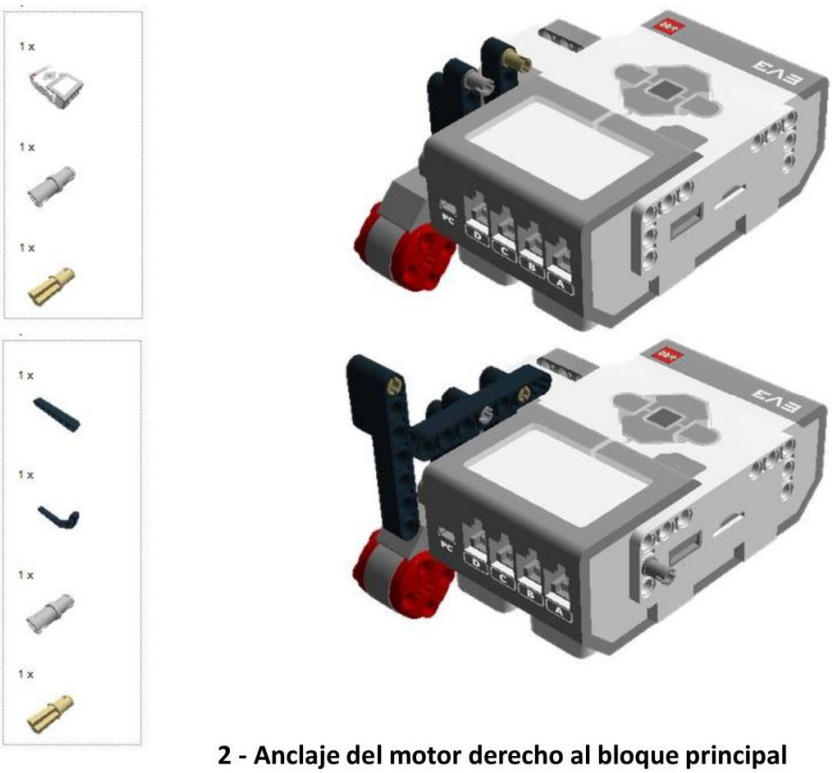
ANGLE ELEMENT, 0 DEGREES [1]		32013	21 - Bright Red	2
CROSS BLOCK 3M		42003	21 - Bright Red	2
CROSS AXLE 9M		60485	194 - Medium Stone Grey	1
CROSS BLOCK 3X2		63869	194 - Medium Stone Grey	1
CROSS BLOCK/FORM 2X2X2		92907	194 - Medium Stone Grey	1

III - Ensamble del robot DVC-Lin3

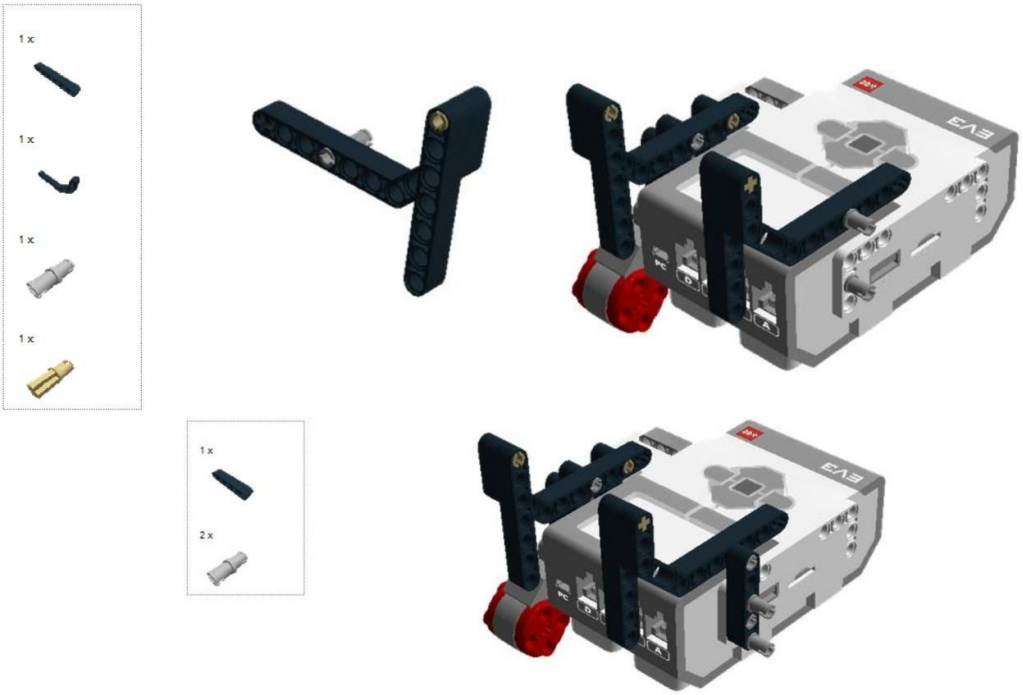
A continuación se detalla, con una guía paso a paso, el proceso de ensamble del robot propuesto en la tesis.



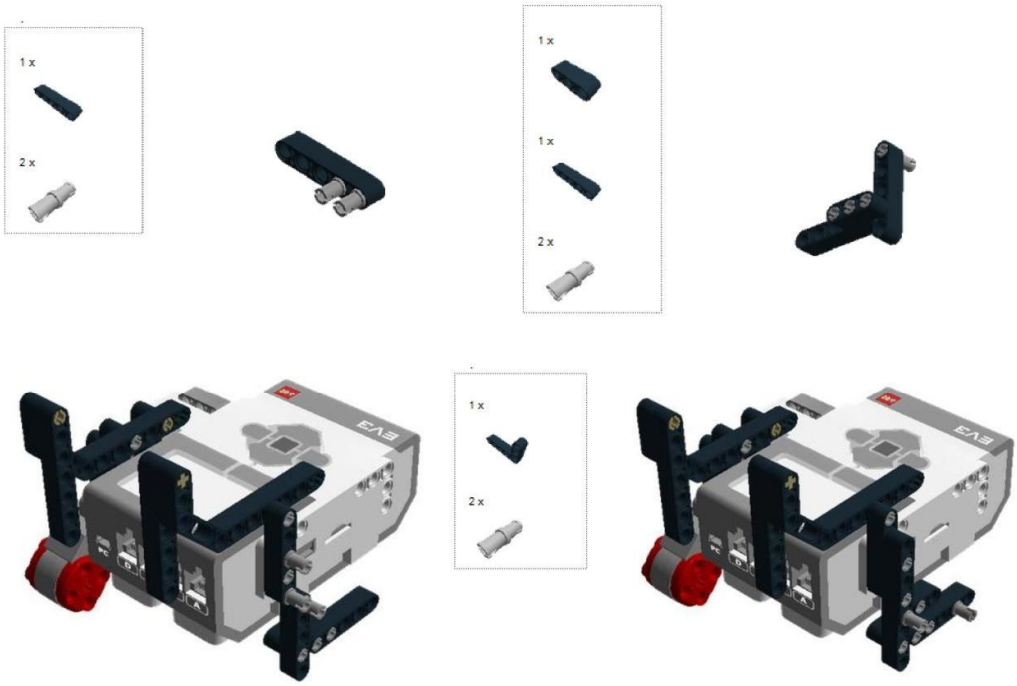
1 - Armado del motor principal derecho



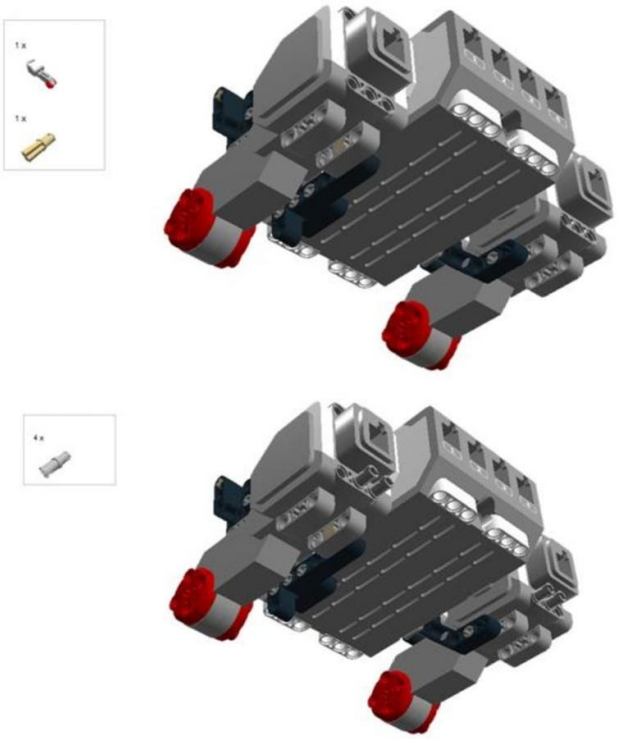
2 - Anclaje del motor derecho al bloque principal



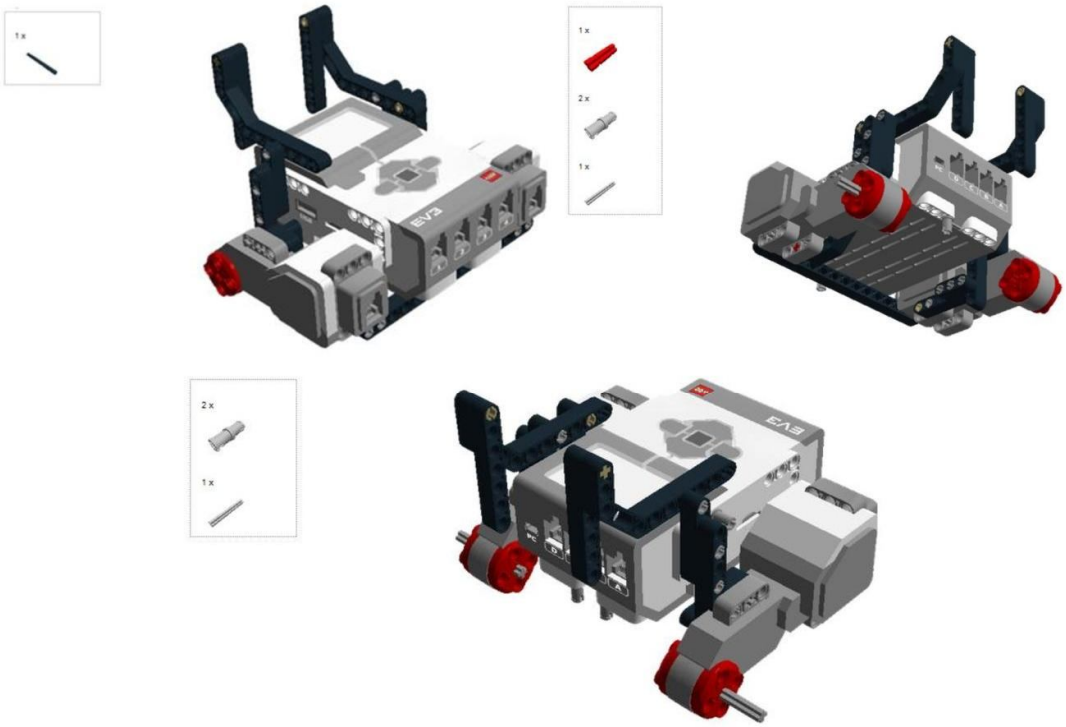
3 - Sujetadores del sensor infrarrojo



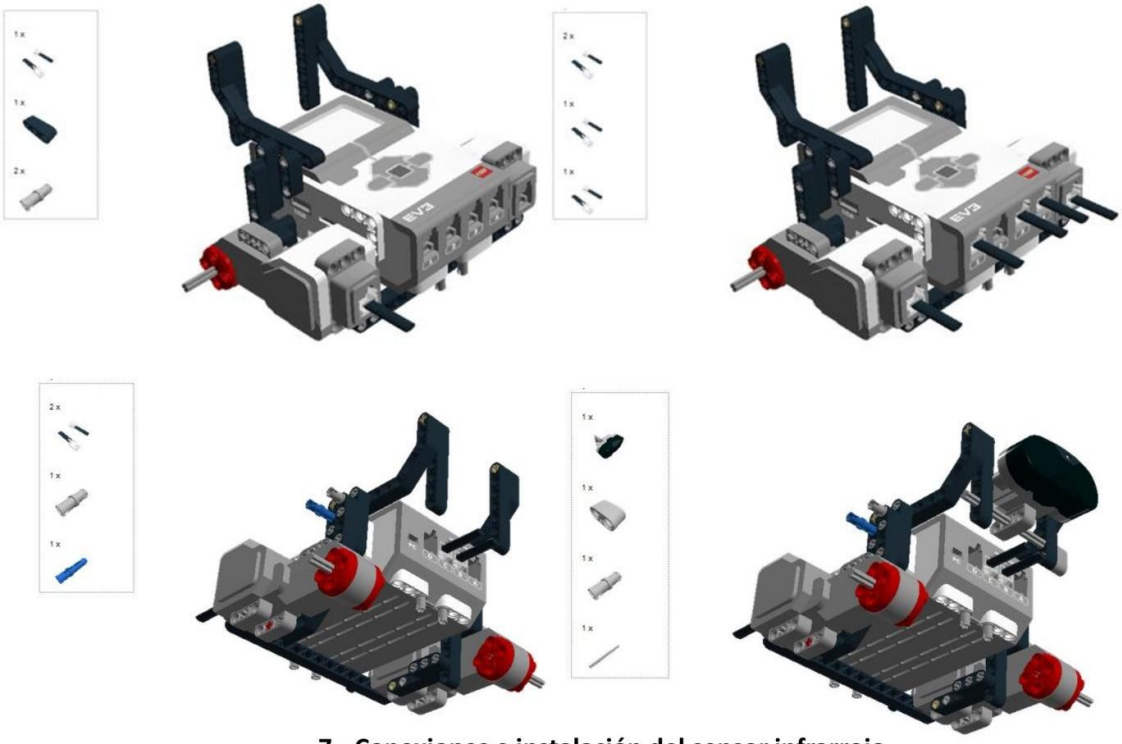
4 - Anclaje motor principal izquierdo



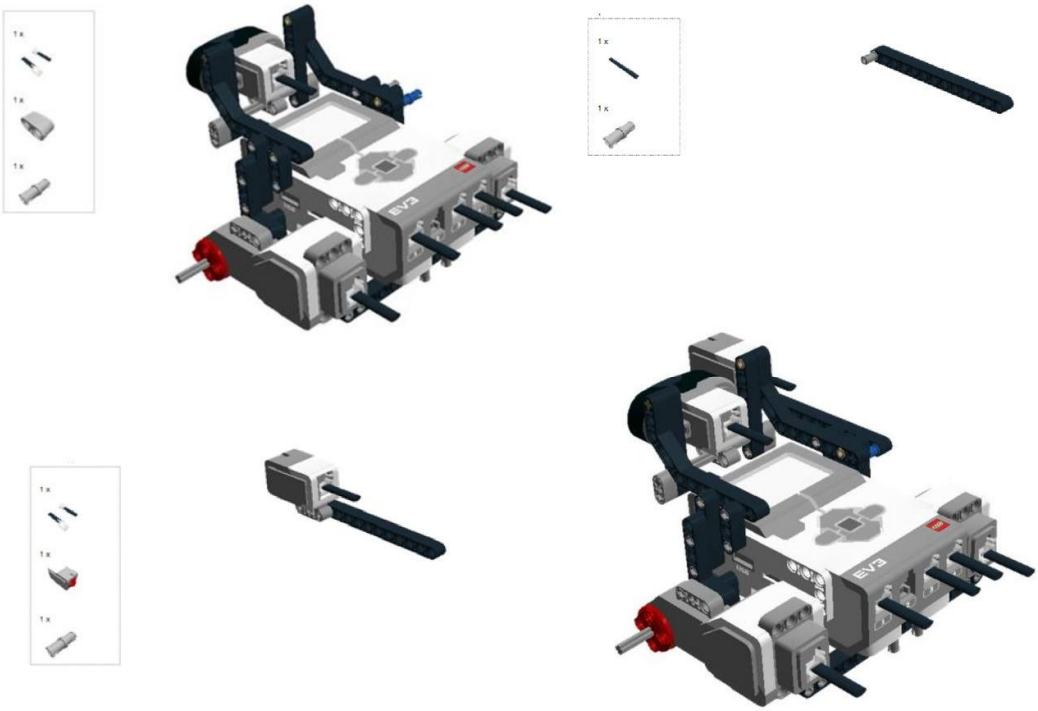
5 - Soportes inferiores del bloque principal



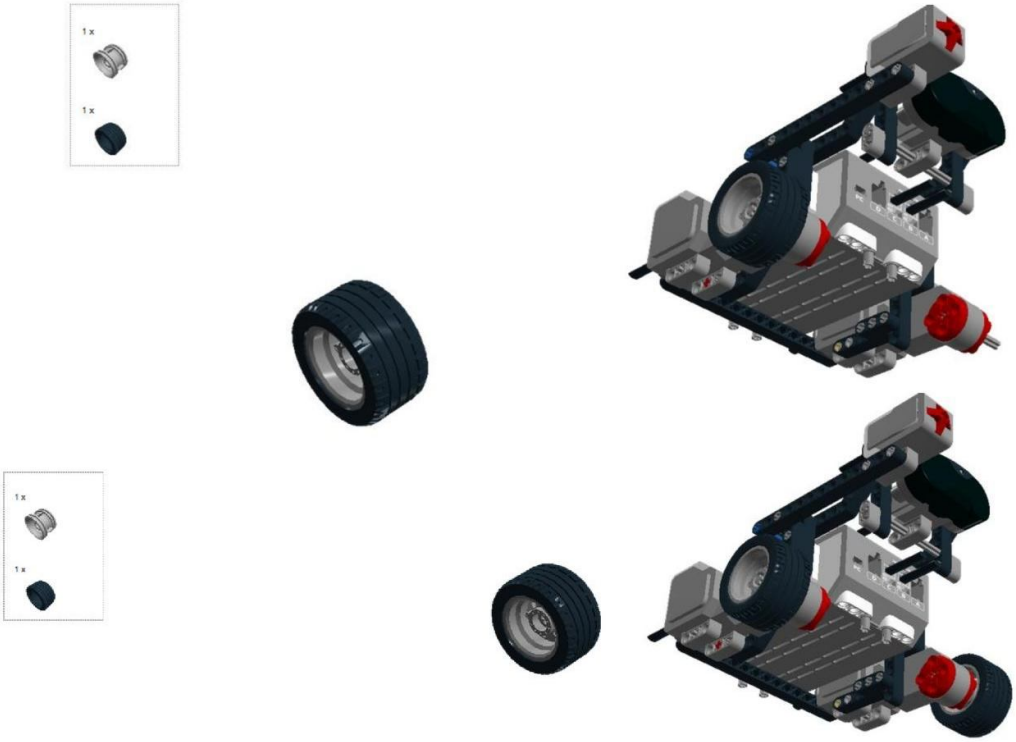
6 - Conexiones fijas inferiores y soporte para las ruedas principales



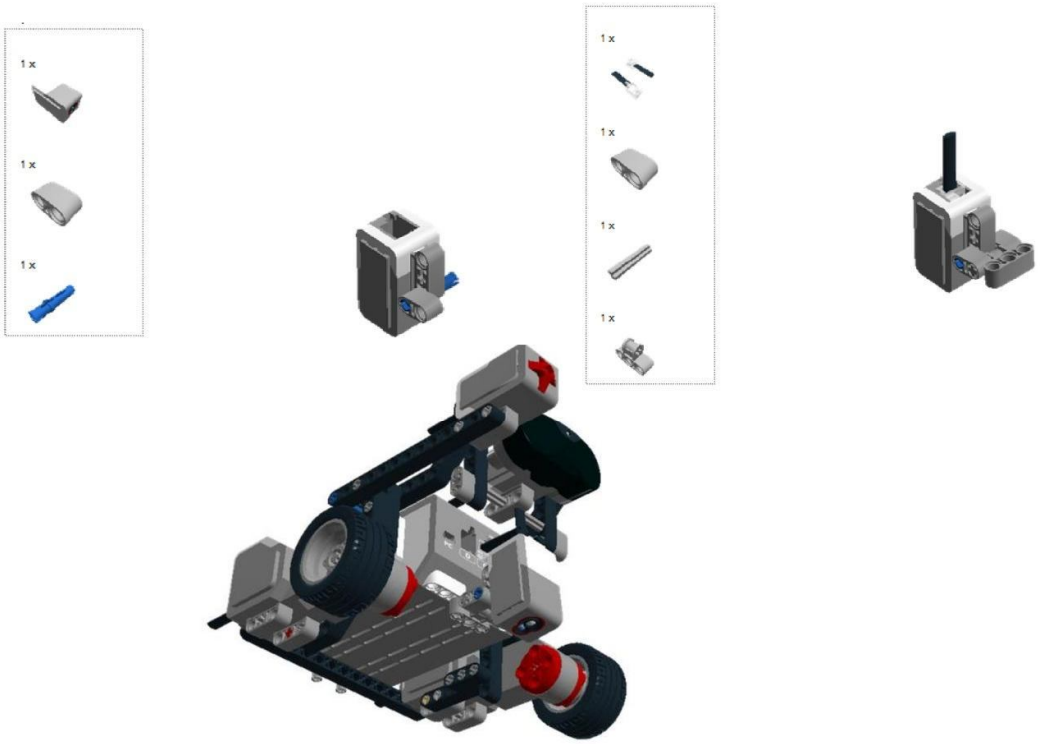
7 - Conexiones e instalación del sensor infrarrojo



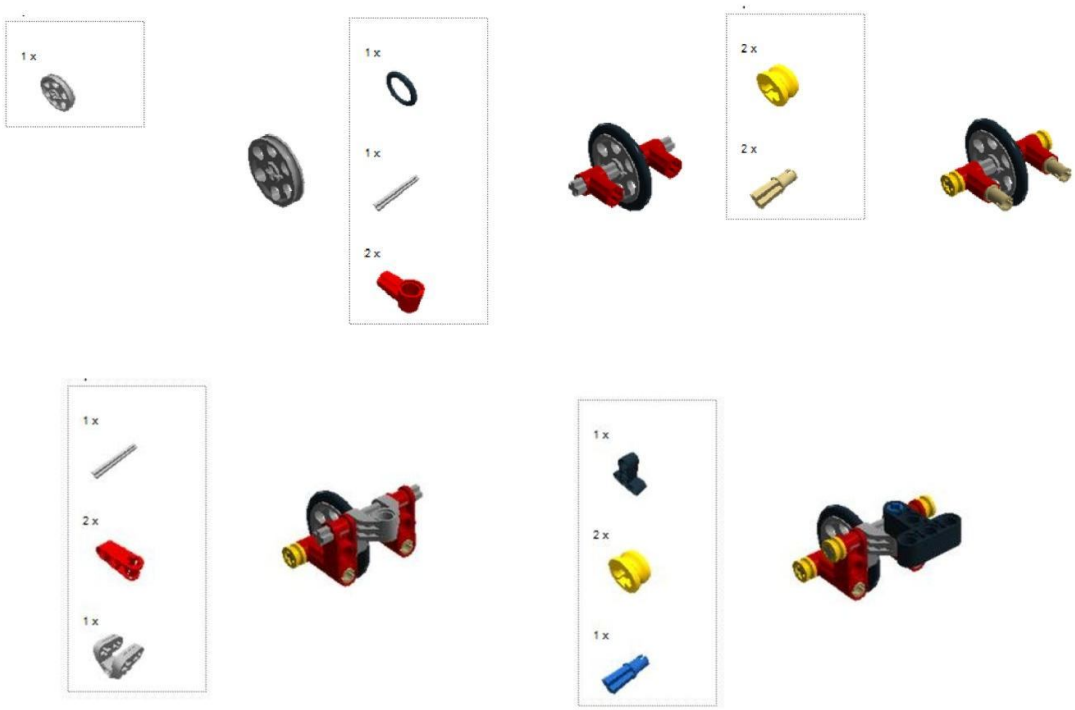
8 - Ensamble del sensor táctil y conexiones restantes



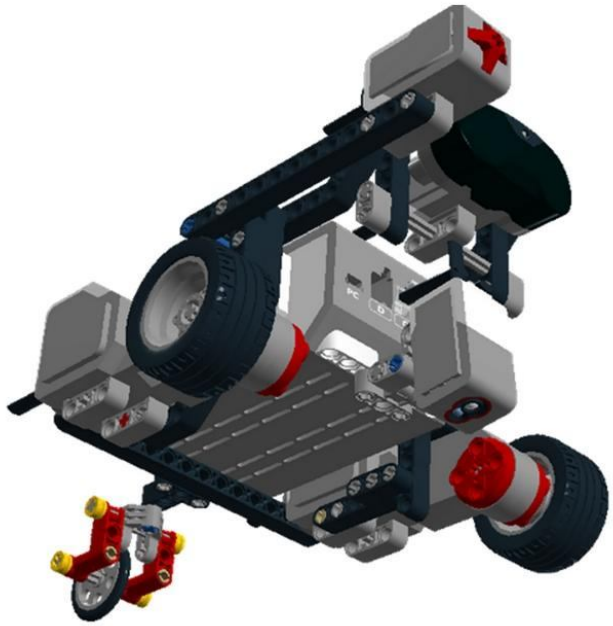
9 - Ruedas principales izquierda y derecha



10 - Instalación del sensor de color



11 - Ensamble de rueda de soporte trasera



Robot terminado: “DVC-Lin3”