Interfaz de programación visual como herramienta educativa para el desarrollo de competencias en ciencia y tecnología por parte de niños, jóvenes y educadores

Daniel Andrés HERNÁNDEZ LONDOÑO

Trabajo de grado para optar al título de Ingeniero Electrónico

Director
Alejandro Martínez
Ingeniero Electrónico

Universidad Pontificia Bolivariana
Escuela de Ingenierías
Ingeniería Electrónica
Programa
Medellín
2016

Declaración de originalidad

08/03/2016

Daniel Andrés Hernández Londoño

Declaro que este proyecto de grado no ha sido presentado para optar a un título, ya sea en igual forma o con variaciones, en esta o cualquier otra universidad, y que el material presentado es de mi autoría.

Dedicatoria

A mis tres madres hasta el último instante

Agradecimiento

Un agradecimiento eterno a la familia que me soporta y me ama, a la mujer que intenta hacer de mí un hombre simple y tranquilo. A los tres maestros que me mostraron lo grandioso de mi carrera y la responsabilidad social que conlleva. El agradecimiento general a todos los que hicieron posible la consecución de este trabajo, en especial a Pygmalion Robotics porque me permitió pasar por su crisol y generar cambios significativos en mi vida y en las de los demás.

Contenido

Lista de tablas

Lista de Figuras

Glosario

Apache 2.0: Es una licencia de *software* libre, no requiere la redistribución del código fuente de versiones modificadas. Bajo esta licencia un software debe conservar el aviso de *copyright* y puede ser usado, distribuido y modificado libremente. Requiere adjudicar crédito de la obra e indicar cambios realizados.

Back-end: Es la parte de la aplicación que procesa la entrada desde el front-end, es decir, todo aquello que sucede internamente en el sistema y que probablemente el usuario no perciba fácilmente.

Bindings: Son adaptaciones de una biblioteca para ser usada en un lenguaje diferente al que fue escrita.

Compilar: La compilación es el proceso de traducción de lenguaje de programación escrito a lenguaje máquina. Un compilador es un programa que traduce dicho programa.

Creative Commons Atribution: Es una licencia de software libre, permite compartir, modificar y distribuir incluso comercialmente. El licenciante no puede revocar estas libertades mientras se sigan los términos de licencia.

Flyout: Es el objeto que describe las características de cada bloque generado, funciona como una ventana emergente.

Framework: Es un conjunto estandarizado de conceptos, prácticas, criterios y herramientas que proveen estructuras o esquemas de desarrollo de aplicaciones. Pueden (o no) estar ligados a un lenguaje de programación en particular como el caso de *Ruby on Rails* o *JavaServer Faces*. En general un framework plantea la estructura global de una aplicación y facilita la colaboración y el uso de utilidades, librerías, etc.

Front-end: Son todos los elementos que interactúan con el usuario, es decir, aquellas funcionalidades que puede percibir de la aplicación.

GNU: La Licencia Pública General GNU garantiza a los usuarios finales la libertad de usar, estudiar, distribuir, compartir y modificar el software. El propósito de esta licencia es declarar que un software cubierto por esta licencia es libre y está protegido frente a la apropiación intelectual.

Handler: Es una función que se activa en la ocurrencia de un evento determinado

Innobot: Es un Kit de Educativo de Robótica diseñado por Pygmalion Robotics®. Es compatible con Arduino IDE.

InnobotC: Es un software de programación derivado de Arduino IDE 1.6.x, adaptado para ser compatible con el kit de Robótica Innobot.

Java: Es un lenguaje de programación de propósito general, concurrente, orientado a objetos, es ideal para el desarrollo de aplicaciones multiplataforma ya que, el código se ejecuta en una plataforma y no tiene que ser recompilado para correr en otra.

JavaScript: Es un lenguaje de programación orientado a objetos, basado en prototipos, imperativo y dinámico. Se utiliza del lado del cliente como parte de un navegador web, permitiendo mejorar la interfaz de usuario o generar páginas web dinámicas.

Python: Es un lenguaje de programación que soporta orientación a objetos, programación imperativa o programación funcional. Es un lenguaje interpretado y multiplataforma.

Renderización: Es un término que se utiliza para referirse al proceso de generación de imágenes o videos, mediante cálculo de iluminación o modelado 2D y 3D. En este trabajo se hace referencia al rederizado de los bloques y objetos bidimensionales generados a través del cálculo de polígonos, arcos, segmentos, etc.

Robótica: En general la robótica, es la ciencia enfocada al diseño e implementación de máquinas capaces de emular el comportamiento de un ser vivo, combinando diferentes disciplinas como la mecánica, electrónica, informática, inteligencia artificial y control para el diseño, construcción, operación, disposición estructural, manufactura y aplicación de la entidad virtual o mecánica artificial denominada robot.

SCRUM: Es un proceso en el que se aplican un conjunto de buenas prácticas para el desarrollo colaborativo, en equipo, con la intención de generar el mejor resultado posible de un proyecto. En un principio, SCRUM se ideó para el desarrollo de software y se considera una metodología ágil donde se realizan entregas parciales, funcionales y regulares del producto final, es ideal para aquellos proyectos donde los requisitos son cambiantes o poco definidos.

STEAM (science, technology, engineering, art, mathematics): Tendencia educativa mundialmente reconocida, donde se pretende profundizar en áreas de ciencia tecnología, ingeniería, artes y matemáticas a través de las nuevas tecnologías de la información y la comunicación, la robótica educativa o la narrativa. Contribuyendo a la consecución de mayor competitividad y por ende una mayor prosperidad económica, reflejándose como un claro índice de la capacidad de un país para mantener un crecimiento sostenido. (Gonzalez B. & Kuenzi J., 2012)

TIC: Las tecnologías de la información y la comunicación, son el conjunto de recursos necesarios para manipular la información.

Toolbox: Es el tablero ubicado al costado izquierdo de la interfaz, contiene todas las categorías que agrupan los diferentes bloques generados,

Workspace: Es el espacio de trabajo donde se generan y disponen los bloques, en este espacio el usuario puede manipular los bloques, realizando conexiones o desconexiones.

Resumen

Entre los problemas más críticos de la educación mundial podemos encontrar la carencia de herramientas metodológicamente efectivas que permitan desarrollar en los niños y niñas conceptos de ciencia, tecnología, ingeniería, artes y matemáticas de forma divertida y en contexto con el entorno en el que se desenvuelven. Entender que el país requiere evolucionar para adaptarse a la sociedad del conocimiento y por lo tanto requiere más y mejores ingenieros, técnicos, médicos, maestros, investigadores o científicos, que sean altamente cualificados que posean una visión social e innovadora para generar cambios significativos. Las estrategias que se implementan para lograr este cometido, se alinean a la tendencia educativa STEAM (por sus siglas en inglés: Science, Technology, Engineering, Arts and Mathematics). Se quiere desarrollar una (1) interfaz de programación visual, cuyo objetivo principal es fomentar el desarrollo del talento humano para las áreas de Ciencia y Tecnología a través de la robótica educativa, como base de las economías del conocimiento. *Copyright © UPB 2016*

Palabras clave: Interfaz, Programación, Educación, Economías del Conocimiento, Robótica, STEAM.

Abstract

Among the most critical global educational issues, the deficiency in technological tools has been identified in order to develop concepts of science, technology, engineering, arts and math for children in a fun way, in context with the ambient in which they evolve. It should be realized this country needs to embrace the knowledge society therefore, more skilled engineers, technicians, researchers, teachers, doctors or scientists are required so people have a social and innovative vision to generate significant changes. The new strategies to deal with this matter are aligned to STEM (Science, Technology, Engineering, Arts and Mathematics) educational tendency. The need to develop a visual programming interface which main objective is to promote the human talent development in science and technology subjects through educational robotics, as the basis of knowledge economies was identified.

Keywords: Interface, Programming, Education, Knowledge Economy, Robotics, STEAM.

INTRODUCCIÓN

Una de las necesidades más apremiantes de la educación mundial es desarrollar estrategias para que niños y jóvenes se acerquen a la ciencia y la tecnología de una forma divertida. Reflexionando, proponiendo y generando soluciones que se adapten a las necesidades locales y globales que impacten de manera real los desafíos de la sociedad futura. Dichas estrategias deberán conllevar al fortalecimiento de la vocación de nuevos ingenieros, investigadores, maestros y científicos altamente cualificados.

En un artículo publicado, la UNESCO expresa: "El mundo necesita más que nunca las soluciones que aporta la ingeniería para hacer frente a desafíos importantes. Sin embargo, en muchos países se está registrando una disminución del número de jóvenes –y sobre todo de mujeres– que estudian ingeniería.", reafirmando lo propuesto anteriormente (UNESCO, 2010).

Existe un consenso general dentro de la comunidad educativa mundial sobre la necesidad de superar las falencias de la educación convencional basada en transmisión de contenidos, a través de corrientes constructivistas y construccionistas. Estas corrientes integran herramientas metodológicas como la robótica educativa, para el fortalecimiento de la ciencia, tecnología, ingeniería, artes y matemáticas (STEAM por sus siglas en inglés: Science, Technology, Engineering, Arts and Mathematics).

Todas las metodologías orientadas a la innovación para la sociedad del conocimiento están enmarcadas por el término STEAM (RutaN, 2014). La robótica educativa en el país se empieza a posicionar como una alternativa seria para apuntalar la educación. En ésta intervienen muchas áreas de conocimiento, generando interrogantes desde muchos frentes para acercar a los estudiantes a la resolución de problemas basados en la observación, la experimentación y la investigación, apropiándose de diferentes herramientas tecnológicas que hasta el momento eran desconocidas, ineficaces o inexistentes. La robótica educativa permite que los estudiantes imaginen, creen y controlen dispositivos reales, visualizando fenómenos físicos y químicos, aplicando conceptos matemáticos, electrónicos y computacionales, descubriendo la programación, etc.

Hoy la tecnología permite utilizar nuevas instancias para fomentar el aprendizaje y talento de forma dinámica y particular. Empresas como LEGO, VEX y ROBOTIS, se han posicionado en el campo de la robótica educativa, ofreciendo soluciones de formación completas en una variedad de áreas curriculares, apuntando a un entorno competitivo que aumente la motivación y deseo de tener éxito a través de habilidades y conocimientos que involucren las cinco disciplinas de STEAM.

A nivel mundial se realizan y promueven diferentes eventos, donde los estudiantes no solo tienen experiencias de vida valiosas y divertidas, si no que permiten desarrollar todo su potencial a nivel cognitivo. Un ejemplo de esto es el Robo RAVE International, un evento que se celebra anualmente en Albuquerque, Estados Unidos,

donde participan delegaciones de Colombia, Estados Unidos, China, México, Argentina, República Checa, Nigeria, Japón, España, Filipinas, entre otros.

En Medellín encontramos a Pygmalion, una empresa , adscrita a la Fundación Parque Tecnológico del Software de Antioquia – ParqueSoft, líder en la robótica educativa en el país basados en las metodologías STEAM, donde se ha diseñado, construido y aplicado diferentes productos y programas educativos para potenciar y acelerar la construcción inevitable de la sociedad del conocimiento en Colombia, entre los cuales se encuentran programas de Jornada Única y Complementaria en los Municipios de Envigado y Medellín, respectivamente.

Entre las herramientas que Pygmalion ha estado desarrollando se destaca una plataforma de programación que integre la forma escrita convencional con nuevos métodos visuales como la programación por bloques, permitiendo así acercar a niños, jóvenes y adultos a conocimientos básicos computacionales, matemáticos y tecnológicos, enmarcados en las metodologías pedagógicas constructivista y construccionista en lineamiento a la tendencia educativa STEAM.

1. PLANTEAMIENTO DEL PROBLEMA

En la década de los 90's, la National Science Foundation (NFS, por sus siglas en inglés), comenzó a utilizar el término "SMET", para referirse a las ciencias, matemáticas, la ingeniería y la tecnología. Dicho término mutaría hasta convertirse en STEM, acrónimo en inglés que hace referencia a las mismas áreas, llegando a posicionarse como una metadisciplina basada en conocimientos de áreas integradas interdisciplinariamente (Sanders, STEM, STEM Education, STEMmania, 2009).

Fue en 2011, cuando las organizaciones NFS y National Research Council (NRC, por sus siglas en inglés), consideraron fundamentales éstas disciplinas para el desarrollo de una sociedad del conocimiento, ya que estás contribuyen a generar riqueza y prosperidad económica en el futuro, siendo un claro indicador de la capacidad de un país para sostener un crecimiento continuo (B. Gonzalez & J. Kuenzi, Science, Technology, Engineering, and Mathematics (STEM) Education: A Primer, 2012).

Existe una tendencia hacia el área de la informática y de servicios que no es ajena al panorama mundial y a la apuesta del Gobierno Nacional en materia de políticas públicas para fortalecer la educación técnica, tecnológica y profesional, acorde a las necesidades del sector productivo y el desarrollo nacional con el avance de la Ciencia y la Tecnología. Esto se puede evidenciar claramente en un diagnóstico en el Conpes 3360 de junio del 2005, donde se propone la necesidad de impulsar la educación técnica y tecnológica, con el apoyo de inversión para el fortalecimiento de las redes y alianzas estratégicas en educación y desarrollo (Ministerio de Educación Nacional, 2005).

Morrison y otros afirman que: "La educación STEM es un enfoque interdisciplinario para el aprendizaje, en donde los conceptos académicos complejos, junto con las lecciones de la vida real de

cómo los estudiantes aplican la ciencia, la tecnología, la ingeniería y las matemáticas que se da en contextos que hacen conexiones entre la escuela, la comunidad, el trabajo y la empresa global, que permita el desarrollo de las competencias STEM y con ella la capacidad de competir en la nueva economía" (Tsupros, Koher, & Hallinen, 2009).

En la última década investigaciones acerca de las artes y la música en la educación, indican que el arte influye positivamente sobre la personalidad de los estudiantes, posibilitando una mejor adaptación a determinados ambientes, beneficiando el aprendizaje significativo. Es por esto que el término STEM ahora no sólo abarca las áreas de Ciencia, Tecnología, Ingeniería y Matemáticas, si no que involucra factores diferenciadores e innovadores hasta concluirse que la creatividad y la innovación del arte es fundamental en todos los procesos de aprendizaje. Esto se traduce en un cambio sustancial del término STEM, donde al incluir el ámbito artístico pasa a ser STEAM (Vásquez Giraldo, 2014).

Muchos teóricos coinciden en que uno de los enfoques importantes de la tendencia STEM es el Pensamiento Computacional, Jeannette Wing, lo define como: "procesos de pensamiento involucrados en formular problemas y encontrar sus soluciones de manera que las soluciones estén representadas de forma tal , que puedan llevarse a cabo efectivamente por un agente que procesa información".

"Es una especie de pensamiento analítico, y que comparte con el pensamiento matemático para resolver problemas, con la ingeniería para el modelado y diseño, limitado por el mundo real, y con el pensamiento científico para la comprensión computacional, la inteligencia, la mente y el comportamiento humano. La esencia del pensamiento computacional es la abstracción que se puede automatizar, que es de lo que trata la informática". (M. Wing, 2006)

Con base a lo anterior, la robótica educativa es una herramienta que no solo facilita el pensamiento computacional, si no que involucra otras habilidades y conocimientos en diversas áreas de investigación y aprendizaje en la educación STEM.

La organización Ruta N y la Secretaría de Educación de Medellín presentaron a las instituciones de Medellín el programa de Innobótica, cuyo objetivo es promover el talento humano con vocación al desarrollo profesional en áreas necesarias para la generación de economías del conocimiento desde edades tempranas, a través de la robótica educativa, la electrónica, transmedia y domótica en la modalidad de jornada complementaria. Al final del proceso los estudiantes proponen una solución al entorno en el marco de ciudades inteligentes.

La secretaría de educación de Envigado ha integrado al nodo científico curricular que abarca tecnología, ciencias naturales, física y matemáticas, la Tecnorobótica, como una apuesta por relacionar diversas áreas del conocimiento y generar en el estudiante una visión mucho más completa. En este proceso la robótica educativa es nuevamente el agente que permite profundizar en múltiples áreas interdisciplinares como la mecánica, la electrónica, la química, etc.

Entre las capacidades que un estudiante debe adquirir, se encuentran el pensamiento sistémico y la adaptación flexible a la tecnología y a los sistemas de información, esto se logra impulsando la ciencia computacional y recurriendo a herramientas de enseñanza no convencionales como la programación por bloques y la sensórica.

En febrero del 2013, se lanzó una campaña que pretende acercar a los niños y jóvenes a la programación y a la ciencia computacional en general, esta campaña es liderada por los fundadores de Facebook, Twitter y Microsoft, en conjunto con la fundación sin ánimo de lucro code.org. Esta campaña se basa en una serie de videos, juegos y clases interactivas, certificables para niños, jóvenes y maestros, que muestran lo divertido de la programación y la gran cantidad de contenido y campos de acción en los que está inmersa. Esta estrategia además pretende escalar la demanda de ingenieros en Estados Unidos. (Guynn, 2013)

En el mercado se pueden encontrar diferentes plataformas de programación orientadas a la robótica educativa, algunos ejemplos como el EV3: Software de programación diseñado por LEGO para la automatización del robot educativo LEGO Mindstorms, ROBOTC y MODKIT diseñadas para el robot educativo VEX. y minBloq diseñada para AERbot, integran la programación por bloques, la descripción grafica de hardware y las simulación de rutinas de código para desarrollar en los estudiantes el pensamiento computacional.

La plataforma que se construirá será accesible a instituciones educativas, docentes y estudiantes, integrará la programación por bloques y la simulación de diversas rutinas en tiempo real, será una plataforma orientada a la ciencia y la tecnología, donde uno de sus componentes más importantes es la robótica educativa.

1.1. Impacto Esperado

En el ámbito social: Fortalecimiento de las estrategias metodológicas curriculares y extracurriculares de los diferentes actores de sistema educativo, entes territoriales e instituciones educativas, para potenciar la afición por la STEAM a través de la robótica educativa cómo mecanismo de apropiación social de la Ciencia y la Tecnología, ofreciendo así alternativas de vocación en dichas áreas.

En el ámbito ambiental: En la medida que la herramienta sea efectiva metodológicamente, los estudiantes se atreverán a resolver problemas, satisfacer necesidades y aprovechar oportunidades por medio de proyectos relacionados con STEAM para mejoramiento de la calidad de vida.

En el ámbito científico y académico: Interrelación con universidades, entidades, áreas del conocimiento y metodologías que pueden enriquecer y complementar el trabajo de la triada Universidad-Empresa-Estado.

Los resultados que se obtengan aportarán al desarrollo del conocimiento pues se diseñará una interfaz de programación visual de robótica educativa en el país basada en las metodologías

STEAM que se articulará a un sistema de aprendizaje integral para niños, jóvenes y educadores.

La metodología desarrollada en este estudio podrá ser aplicada a otros proyectos de características similares, como una herramienta que permitirá involucrar a los niños, jóvenes y adultos en la construcción de una sociedad del conocimiento.

2. OBJETIVOS

2.1. Objetivo General

Desarrollar una (1) Interfaz de programación, como herramienta educativa para el desarrollo de competencias en ciencia y tecnología por parte de niños, jóvenes y educadores.

2.2. Objetivos Específicos

- Analizar la información disponible en diversas fuentes para conocer de manera general el estado del arte de la información.
- Indagar por las corrientes constructivistas y construccionistas para entender las necesidades metodológicas que debe suplir la herramienta a desarrollar.
- Determinar el ciclo de vida del sistema utilizando el modelo incremental.
- Implementar la metodología de desarrollo incremental.
- Realizar el diseño preliminar del sistema transformando los requisitos en la arquitectura del software.
- Realizar el diseño detallado del sistema ocupándose del refinamiento y representación arquitectónica que lleva a la representación arquitectónica y algorítmica del software.
- Desarrollar e implementar una lista de control del sistema que incluya el historial de tareas que se deben realizar, funcionalidades para ser implementadas y áreas de rediseño o mantenimiento de cada una de las versiones de la interfaz.
- Realizar cinco (5) incrementos validados por pruebas de aceptación del usuario (UAT por sus siglas en inglés: User Acceptance Testing), para asegurar el comportamiento del sistema y verificar el alcance de la interfaz.
- Compilar el programa para diferentes sistemas operativos (Windows y Linux).
- Implementar un sistema de comunicación y transferencia de datos entre el software y la unidad de control del robot Innobot.

3. MARCO TEÓRICO

La característica más importante de la civilización moderna reside en la rapidez con la que se producen los cambios. La dinámica de la economía y el mercado, incluyendo las calificaciones profesionales que exige, evoluciona vertiginosamente. Este

impacto se denomina "*shock* del futuro". Debido a la gran dificultad que tienen las sociedades para procesar el cambio.

El *shock* del futuro es un término acuñado por Alvin Toffler quien dice que: Para enfrentar el futuro, al menos en la medida de nuestras posibilidades, es más importante ser creativo y perceptivo que estar cien por ciento en lo "cierto". No es necesario que una teoría sea "cierta" para que sea de gran utilidad. Incluso los errores pueden ser útiles. "Los mapamundi que dibujaban los cartógrafos de la Edad Media eran tan irremediablemente imprecisos y estaban tan plagados de errores que hoy en día casi podríamos decir que nos producen ternura… Y, sin embargo, los grandes exploradores de la época jamás habrían descubierto el Nuevo Mundo sin ellos". (Toffler, 1990)

Según Perelman, la división clásica de la economía comprendía tres sectores fundamentales: agricultura, industria y servicios. No obstante, se debe contemplar también el sector del conocimiento, que involucra a los llamados "trabajadores del conocimiento" y la visión de un entorno cada vez más automatizado, donde la tecnología es cada vez más "inteligente" y el trabajo mental se aleja paulatinamente del mero procesamiento de información, para centrarse en la creación de información y conocimientos nuevos, para comunicarlos, intercambiarlos y compartirlos. El trabajo del conocimiento no constituye simplemente un nuevo sector, sino un eje trasversal, factor común presente en todas las actividades económicas contemporáneas. (Perelman, 1992)

La vida en la sociedad del conocimiento, requiere que los individuos se preparen para tomar decisiones responsables, frente a situaciones nuevas e inesperadas, aprendiendo constantemente a lo largo de la vida a analizar, sintetizar, evaluar, canalizar y presentar información con pensamiento crítico para hacer un uso productivo de la tecnología.

Es esencial que los niños, jóvenes y adultos del mundo actual cuenten al menos con una noción general de cómo utilizar las herramientas tecnológicas de su entorno, y para ello, los educadores deben ayudar a desarrollar una visión del futuro.

Un problema básico de la enseñanza es conocer como las personas representan mentalmente su conocimiento acerca de lo que les rodea, como se adaptan en los diferentes contextos. Y para ello, se han planteado diversas estrategias, como el uso de la narrativa y los ambientes de aprendizaje computarizados.

Los ambientes de aprendizaje computarizados se han posicionado como un complemento importantísimo en el aula, puesto que abren diferentes perspectivas a los conceptos de espacio y tiempo, modificando la relación profesor – estudiante y las formas de comunicación y actuación. Esta herramienta, fue concebida como todos aquellos elementos físico - sensoriales, tales como la luz, el color, el sonido, el espacio, el mobiliario, etc; que caracterizan el lugar donde un estudiante ha de realizar su aprendizaje. (Husén & Postlethwaite, 1989). Sin embargo, en la actualidad hay diversas formas de concebir un ambiente de aprendizaje en la educación formal (Moreno, Chan, Pérez, Ortiz, & Viesca, 1998), ya que se

contemplan no solo los espacios físicos y los medios, sino también los elementos básicos del diseño instruccional, utilizando todos los recursos tecnológicos disponibles para crear situaciones favorables en el proceso de aprendizaje y construcción de conocimiento. En particular, un software educativo está destinado a la enseñanza y aprendizaje autónomo y en conjunto con otras herramientas tecnológicas se convierten en una de las formas más efectivas de crear ambientes de aprendizaje.

Un sistema capaz de concebir ambientes de aprendizaje no puede carecer de fundamento pedagógico ya que como Shulman lo propone: la pedagogía no debe estar descontextualizada de la materia que se imparte y, por lo tanto, debe estar impregnada y condicionada por ella. Hay que conocer lo que se enseña y cómo debe ser enseñado. (Gudmundsdottir & Shulman, 1987)

3.1. Del Constructivismo al Construccionismo

Mario Carretero, define el constructivismo como "la idea que se mantiene de que el individuo, tanto en los aspectos cognitivos y sociales del comportamiento como en los afectivos, no es un mero producto del ambiente ni un simple resultado de sus disposiciones internas, sino una construcción propia que se va produciendo día a día como resultado de la interacción entre esos dos factores. En consecuencia, según la posesión del constructivismo, el conocimiento no es una copia fiel de la realidad, sino una construcción del ser humano. ¿Con qué instrumentos realiza la persona dicha construcción?, fundamentalmente con los esquemas que ya posee, es decir, con la que ya construyó en su relación con el medio que lo rodea". (Carretero, 1997)

"El constructivismo es una confluencia de diversos enfoques psicológicos que enfatizan la existencia y prevalencia en los sujetos cognoscentes de procesos activos en la construcción del conocimiento, los cuales permiten explicar la génesis del comportamiento y el aprendizaje. Se afirma que el conocimiento no se recibe pasivamente ni es copia fiel del medio". (Arceo Barriga & Hernández Rojas, 2004)

El aprendizaje constructivo, supone entonces una construcción que se efectúa por medio del proceso mental que implica adquirir conocimiento nuevo, dicho proceso además desarrolla competencias que le permitirán al estudiante aplicar lo ya aprendido a una nueva situación y responder a las demandas del entorno. El constructivismo es flexible y dinámico, se adapta a los contextos y se nutre de la exposición a la información a la que se enfrenta el sujeto.

Por otro lado, Seymour Papert destaca la importancia de la acción para producir aprendizaje, el conocimiento construido por el propio sujeto a través de la acción. Esta teoría se conoce como construccionismo y se enfoca en la construcción de mundos contextualizados.

El aprendizaje construccionista involucra a los estudiantes a través de la experimentación, la proposición de ideas y la elaboración de objetos sociales. La enseñanza se sustituye por la asistencia al estudiante a lo largo de sus propios descubrimientos y

construcciones que le permitan comprender, entender y transformar el entorno.

En relación con el constructivismo, Papert sostiene que: "Tomamos de las teorías constructivistas de la psicología el enfoque de que el aprendizaje es mucho más una reconstrucción que una transmisión de conocimientos. A continuación, extendemos la idea de materiales manipulables a la idea de que el aprendizaje es más eficaz cuando es parte de una actividad que el sujeto experimenta como la construcción de un producto significativo". (Papert, 1986)

3.2. Metodología de desarrollo

Las metodologías de desarrollo de software, hacen referencia al marco de trabajo que permite estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Una metodología en el desarrollo de software se denomina *framework*, que en ingles significa infraestructura, armazón o marco.

Un framework consiste en una filosofía de desarrollo de programas de computación con un enfoque del proceso de desarrollo de software determinado y un conjunto de herramientas, modelos y métodos que facilitan el diseño, implementación e implantación del software. (Green & DiCaterino, 1998)

3.2.1. Ciclo de vida del sistema

El desarrollo del ciclo de vida de software sirve para establecer las distintas etapas y estados por los que debe pasar el sistema, desde la concepción inicial, pasando por el desarrollo, puesta en marcha, mantenimiento y retirada del producto. Es necesario decantarse por un modelo de ciclo de vida que describa las fases y el orden en se ejecutarán.

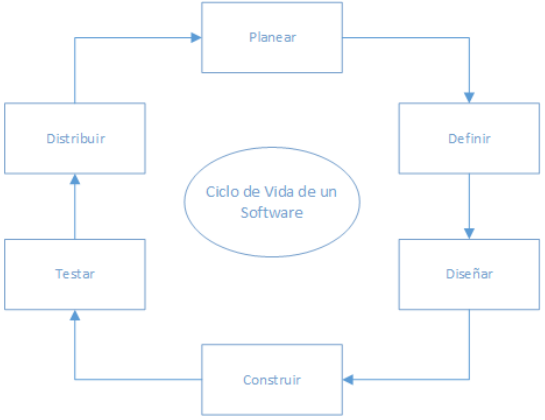


Figura 1. Ciclo de Vida de un Software

Existen diversos modelos que definen el desarrollo del ciclo de vida de un software. A continuación se realiza una introducción de los modelos convencionales:

3.2.1.1. *Modelo en Cascada*

El modelo en cascada ilustra el proceso de desarrollo de software en secuencia lineal; es decir, que cualquier fase de desarrollo empieza solo cuando la fase previa se haya completado. Las fases no se solapan.

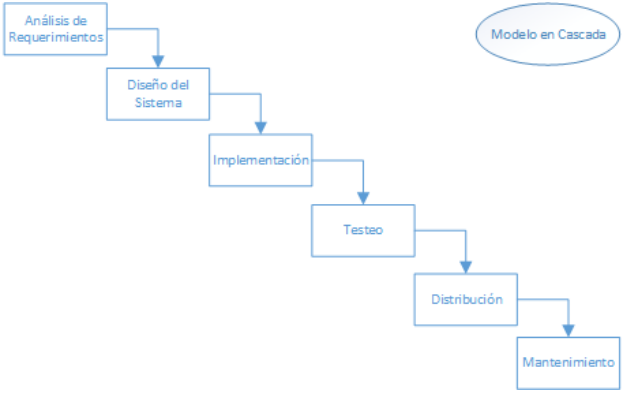


Figura 2. Modelo en Cascada

3.2.1.2. *Modelo Incremental e Iterativo*

Este modelo se trata de realizar implementaciones simples de los requerimientos iterativamente, generando versiones funcionales, hasta que todo el sistema se haya implementado. En cada incremento se realizan modificaciones de diseño y se agregan nuevas funcionalidades, hasta generar una versión estable y funcional del sistema aprobada e instaurada.

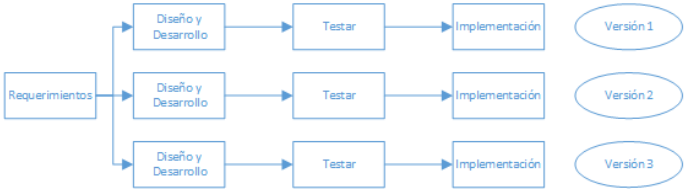


Figura 3. Modelo Incremental e Iterativo

3.2.1.3. *Modelo en Espiral*

El modelo en espiral tiene cuatro fases. Un proyecto debe pasar a través de estas cuatro fases en un cierto número de iteraciones. El conjunto de iteraciones realizadas se puede representar en forma de espiral.

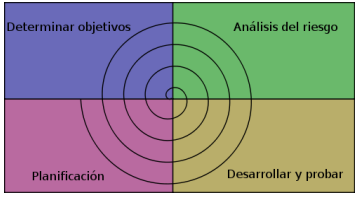


Figura 4. Modelo en Espiral. Tomada de (Wikipedia, 2016)

3.2.1.4. Modelo en V

Este modelo se divide en fases de verificación y fases de validación. Las actividades que tienen correspondencia entre si se realizan en paralelo a lo largo del proyecto.

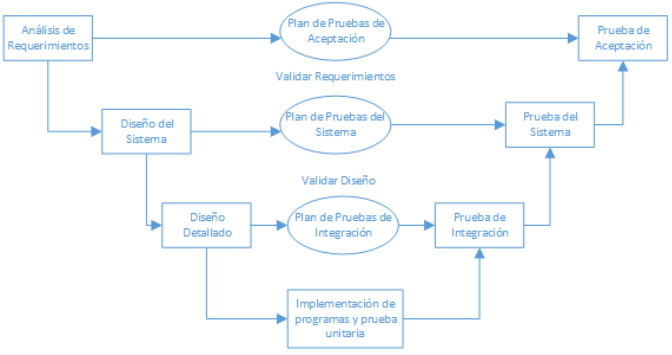


Figura 5. Modelo en V

3.2.1.5. Modelos Ágiles

Los modelos ágiles son caracterizaciones de los modelos convencionales, integrando ciertas prácticas, herramientas, frameworks, filosofías y métodos de desarrollo. En la actualidad es muy común encontrar metodologías como SCRUM, Crystal Clear,

Tabla 1. Comparación de Modelos

Aspectos	*Cascada*	*Incremental*	*Espiral*	*Modelo en V*
Requiere Conocimiento inicial de todos los requerimientos	Si	No	No	Si
Permite Cambios en los requerimientos	No	Si	Si	No es Recomendable
Nivel de Costos	Alto	Medio	Medio	Alto
Nivel de Organización	Alto	Medio	Alto	Alto
Nivel de Flexibilidad	Bajo	Medio	Medio	Medio
Disponibilidad del proyecto en menos tiempo	No	Si	Si	Si
Tipo de Software	Pequeño	Pequeño, grande y mediano	Pequeño, grande y mediano (Complejo)	Mediano
Evaluación de riesgos	Bajo	Medio	Alto	Medio
Índice de errores en proyectos finalizados	Alto	Bajo	Bajo	Medio

> Programación Extrema (XP), entre otras. Haciendo una comparación puntual con los modelos descritos anteriormente se determina que la metodología SCRUM, es adaptable un trabajo de estas dimensiones, debido a que el desarrollo se aborda a través del modelo incremental e iterativo, el solapamiento de las diferentes fases del desarrollo, y la implementación de estrategias de desarrollo reconocidas en la industria del software. Sin embargo, la implementación de una metodología ágil, requiere mucha experiencia, un equipo suficiente y altamente capacitado, usualmente se priorizan los resultados a la documentación.

En este proyecto se propone entonces el modelo de desarrollo incremental e iterativo, que combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa paulatinamente. Para ello, se realizan secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal debe producir un incremento relevante del software.

Este modelo ayuda a generar software operativo de forma rápida, reduce el coste en el cambio de alcance y requisitos. Es más fácil probar, validar y depurar en una iteración más pequeña, es más sencillo gestionar hitos y riesgos. (INTECO, 2009)

3.2.2. Etapas del modelo de desarrollo incremental

3.2.2.1. Etapa de iniciación

> Se determinan los aspectos claves del programa en concordancia con el objetivo general del proyecto. Se determinan los incrementos necesarios para llevar a cabalidad el proyecto y se define en detalle aquellas funcionalidades o servicios que se van a entregar en el primer incremento. Luego, se crea una versión del sistema que tiene como objetivo liberar un producto con el que el usuario pueda interactuar, y por ende, validar y/o retroalimentar el proceso. Esta versión debe proveer una solución suficientemente simple para ser comprendida e implementada fácilmente. En este punto se debe crear una lista de control que se revisará periódicamente. Esta lista debe incluir el historial de tareas que se deben realizar, funcionalidades para ser implementadas y áreas de rediseño o mantenimiento de la solución ya existente. (Sommerville, 2005)

3.2.2.2. Etapa de Iteración

Esta etapa involucra el rediseño, mantenimiento e implementación de una tarea de la lista de control del proyecto y el análisis de la última versión del sistema. El análisis de las iteraciones o incrementos se consigue con la retroalimentación del usuario y el análisis de las diferentes funcionalidades disponibles del sistema. Se debe tener en cuenta el análisis de la estructura, modularidad, usabilidad, confiabilidad, eficiencia y eficacia. (Sommerville, 2005)

3.2.3. Diseño del Sistema

En el diseño del sistema se optará por un modelo de repositorio, partiendo de la premisa de que los subsistemas que forman un sistema deben intercambiar información para que puedan trabajar conjuntamente. Es decir que, todos los datos compartidos se almacenaran en una base de datos central a la que se puede acceder por todos los subsistemas. (Sommerville, 2005)

En el diseño, se optará por una descomposición orientada a objetos, donde los objetos realizarán llamadas a los servicios ofrecidos por otros.

3.3. Usabilidad

Muchos autores han propuesto diversas definiciones de usabilidad, describiendo cada uno de los atributos o factores mediante los que puede ser valorada, dependiendo finalmente del enfoque con el que pretende ser medida. (Folmer & Bosch, 2004)

La definición más extendida y objetiva, es la ofrecida por la ISO, que define usabilidad como el "grado de eficacia, eficiencia y satisfacción con la que usuarios específicos pueden lograr objetivos específicos, en contextos de uso específicos". (International Organization for Standardization, 1998)

La usabilidad entonces se compone de dos atributos fundamentales:

Aquellos atributos cuantificables de forma subjetiva como la satisfacción de uso, medible a través del mecanismo de interrogación a usuario. Y aquellos atributos cuantificables de forma objetiva, que son la eficiencia o número de errores cometidos por el usuario durante la realización de una tarea específica y finalmente el tiempo empleado por el usuario para la consecución de la misma.

Un concepto ligado al de usabilidad es el de accesibilidad, el cual se entiende como la posibilidad de acceso, es decir, que el diseño de la aplicación solo es usable, si posibilita el acceso de todos los usuarios potenciales, sin excluir a aquellos con limitaciones individuales, capacidades diversas, dominio del idioma o limitaciones derivadas del contexto de acceso - software, hardware, ancho de banda, etc. (Hassan, Fernández, & Iazza, 2004)

3.3.1. Método por inspección: evaluación heurística

Este tipo de evaluación la lleva a cabo un grupo reducido de evaluadores que se basan en la experiencia y en la fundamentación de reconocidos principios de usabilidad, apoyándose en guías elaboradas para dicho fin. Entre los principios que se evalúan se encuentran:

- La visibilidad del estado del sistema: el sistema siempre debe informar al usuario acerca de lo que está sucediendo.
- El lenguaje entre sistema y usuario: el sistema debe interactuar con el usuario, sin tecnicismos incomprensibles o mensajes crípticos.
- La libertad y control por parte del usuario: el usuario debe tener el control del sistema y de la forma de interactuar con el mismo.
- Consistencia y estándares: conservar la identidad gráfica y funcionalidad del sistema.
- Reconocimiento del sistema: la visibilidad de las diferentes opciones, enlaces, objetos, etc.

3.3.2. Método de test con usuarios:

Este test se basa en la observación y análisis de usuarios reales antes, durante y después de la interacción con el sistema, este tipo de test debe ser realizado en diferentes etapas del desarrollo de software, ya que cuanto más tiempo se tome en su realización, más costosa resultará la reparación o depuración de errores. Además, permite entender la tendencia del mercado y valorar la efectividad de la solución propuesta con mayor precisión y exactitud.

3.3.3. Test formales y test informales

3.3.3.1. Test Formales

Se realizan en un laboratorio equipado con tecnologías avanzadas de grabación.

Se realiza al final de todo el proceso con el desarrollo completo.

Los sujetos son seleccionados cuidadosamente para ser representativos estadísticamente.

Se conducen como experimentos reales con el objetivo de confirmar o refutar una hipótesis.

3.3.3.2. Test Informales

No es necesario utilizar laboratorios equipados con tecnologías avanzadas de grabación.

Se realiza en diferentes momentos del proceso sobre los prototipos generados.

Se utilizan los sujetos disponibles, procurando que sean representativos estadísticamente.

Se emplean de forma recurrente para hacer visibles las deficiencias de usabilidad y modelar el producto gradualmente.

Jakob Nielsen propuso un test en la década de los 90's. El test ofrece una serie de pautas para conseguir la mayor información posible de un usuario durante su experiencia con el producto.

Se recomienda que este test se ejecute después de una evaluación heurística, puesto que se puede depurar un porcentaje de errores críticos antes de que el usuario utilice la aplicación.

Nielsen observó, haciendo test con multitud de usuarios, que a partir de cierto número de usuarios evaluados, los errores se repetían y no se aportaba más información relevante. Con cinco usuarios es posible encontrar más de un 80% de errores de usabilidad y por lo tanto se recomienda que el número mínimo de usuarios evaluados en un test sea de tres a cinco.

Cada test es diferente y se deben priorizar con anticipación aquellas funcionalidades o tareas que se van a evaluar, recordando que este test sirve para detectar problemas de usabilidad y no para resolverlos.

Se debe crear un guion que describa las tareas que el usuario debe realizar, es usual que el usuario deba realizar un recuento detallado de cada una de las acciones que realiza sobre el sistema en voz alta, incluyendo pensamientos y apreciaciones.

Según la definición de Nielsen "Es un test en el que se le pide al participante que use un sistema mientras piensa continuamente en voz alta, verbalizando sus pensamientos mientras se mueve por la interfaz de usuario". Se debe grabar todo el proceso de testeo. Tanto la interacción con la interfaz como el usuario deben quedar registrados para un

posterior análisis. Es muy importante que el moderador del test no brinde ayuda durante la prueba, de esta forma se evita que el resultado del test se altere.

El criterio para definir el éxito de una tarea incluye, alcanzar ciertos objetivos de la tarea, un número máximo de errores, completar la tarea en un tiempo determinado, etc.

Las tareas se pueden priorizar por frecuencia, vulnerabilidad o por su preponderancia en el sistema. Al priorizar por la frecuencia, se determinan aquellas tareas que se realizan durante el 75% u 80% del tiempo cuando se utiliza la aplicación. La vulnerabilidad permite observar las tareas que pueden incluir defectos de diseño, fácilmente identificables por el equipo de desarrollo. Cuando se prioriza por preponderancia en el sistema se identifican las tareas que tienen efectos desastrosos para el usuario, el producto o la compañía, como la pérdida de datos. (Nielsen, 1993)

4. ADQUISICIÓN DE REQUERIMIENTOS

La adquisición de requisitos es el proceso de recolección de información necesaria para construir el sistema. Este proceso proporciona los lineamientos a seguir y las características del sistema que se deben validar tras cada incremento en el desarrollo. Obtener la cantidad y tipo adecuado de información en el momento apropiado es esencial en el desarrollo de software.

4.1. Revisión de otras interfaces de programación en el mercado

Esta revisión permite entender muchos de los requerimientos básicos del sistema, de acuerdo a las tendencias del mercado.

4.1.1. Scratch

Scratch es un lenguaje de programación educativo y una herramienta multimedia de edición de lenguaje de programación, que se apoya en la corriente constructivista para generar ambientes de aprendizaje en diferentes áreas del conocimiento. Incluye simulaciones, visualizaciones de experimentos, presentaciones, cuentos animados e interactivos y diferentes expresiones artísticas.

Scratch nació como una implementación de un lenguaje reflexivo de programación, orientado a objetos y tipado dinámico llamado Smalltalk. Smalltalk se considera un entorno de objetos o un mundo virtual donde se alojan objetos que se comunican entre sí, mediante el envío de mensajes.

A partir de la versión 2.0 de Scratch, el código se rescribió en Adobe ActionScript, donde se incluyó la construcción de animaciones, datos e interfaces interactivas.

Scratch cuenta con una versión Online que permite guardar y compartir proyectos realizados. (Milonovich, 2013)

4.1.2. Blockly

Google diseñó un lenguaje de programación visual, compuesto por un editor de programación gráfico implementado en JavaScript, que se puede compilar para JavaScript Dart o Python. Los usuarios tienen a su disposición un conjunto de comandos que se pueden combinar como si se tratase de un rompecabezas. Blockly pretende priorizar la lógica y evade la sintaxis de programación que puede resultar engorrosa para principiantes.

Blockly está licenciado bajo Apache 2.0 y ha influenciado diversas herramientas gráficas de codificación como App Inventor, Snap! o inclusive Scratch.

En general Blockly permite arrastrar objetos vectoriales que se traducen en componentes de control, lógica, operaciones matemáticas, texto, etc. Un conjunto de componentes se puede exportar a lenguajes como JavaScript, Dart, Python, C o XML.

Toda la documentación y desarrollo de este proyecto es libre y puede ser modificado o distribuido, siempre y cuando se respeten los términos bajo los que fue licenciado. (Google, 2015)

Neil Fraser, uno de los creadores de Blockly, dice que "cada generación llega a utilizar una interfaz de programación aún de más alto nivel y eventualmente se podría instruir a las computadoras con un lenguaje completamente natural. Momento en el que todo el mundo será capaz de decirle a una computadora lo que debe hacer".

Blockly no se centra en desarrollar archivos ejecutables o aplicaciones de usuario final. Su objetivo principal es que todas las personas, principiantes, expertos, niños, jóvenes o adultos puedan participar en el diseño scripts sencillos o complejos que faciliten el desarrollo de aplicaciones o contenidos.

4.1.3. Visualino

Visualino es una herramienta de programación visual para Arduino, en un entorno similar a Scratch. La compilación de los archivos se realiza

backend y utiliza los recursos de la versión 1.6 de Arduino para compilar las diferentes tarjetas de desarrollo. Su interfaz está basada en Google Blockly.

Visualino es una aplicación libre, multiplataforma y la documentación del proyecto se puede encontrar en español e inglés.

4.1.4. Bitbloq

Este entorno de programación, fue diseñado por una marca española llamada BQ, que se dedica al diseño, venta y distribución de lectores electrónicos, tabletas, teléfonos inteligentes, impresoras 3D y kits de robótica. La empresa matriz es Mundo Reader S.L. Bitbloq se construyó a partir de Google Blockly, pero para el 2016 reescribieron todo el código en JavaScript que genera los objetos vectoriales (SVG), utilizando su propia librería, este proyecto es libre al público pero carece de documentación.

Bitbloq está pensada como una aplicación web que utiliza un web socket y un paquete de instalación de drivers para compilar las diferentes tarjetas de desarrollo asociadas a la empresa. El script generado se puede compartir y guardar en la nube.

4.1.5. *Modkit*

Modkit es una interfaz de programación que permite controlar e interactuar con los kit de robótica desarrollados por VEX Robotics. El ambiente gráfico es similar a Blockly, pero los

Tabla 2. Requerimientos Funcionales del Sistema

1.	Generación de bloques que representen código funcional.
2.	Capacidad de generación de código a través de los bloques generados y manipulados.
3.	Capacidad de traducción de código generado a texto plano codificable.
4.	Implementación de un editor de código de texto plano con capacidad para autocompletar funciones
5.	Capacidad de cargar y compilar el código generado por el editor de código
6.	Capacidad de compilar y cargar código generado por un conjunto lógico de bloques.
7.	Capacidad de compilar y cargar código a diferentes tarjetas de desarrollo.
8.	Capacidad de abrir y guardar un proyecto o trabajo realizado
9.	Capacidad de representar librerías
10.	Capacidad de comunicación serial con la tarjeta de desarrollo programada.

Tabla 3. Comparación funcional de Interfaces con documentación

Aspectos	*Scratch*	*Blockly*	*Bitbloq*	*Visualino*
Lenguaje de Programación	ActionScript (Adobe Flash)	HTML, JavaScript	XML, HTML, JavaScript	HTML, JavaScript
Requerimientos	Chrome 7 Explorer 7 o posteriores, Adobe Flash Player 10.2	Chrome 7, Firefox 4, Internet Explorer 7, Safari 5.0, Opera 11 o superiores	Chrome 7 o superior	Arduino 1.6 o superior instalado
Implementación	Online y portable	Online	Online	Utiliza un Webkit que se ejecuta localmente, es portable
Sistema Operativo	Windows, Linux, OS X	Windows, Linux, OS X	Windows, Linux, OS X	Windows, Linux, OS X
Idioma	40 idiomas	43 idiomas	6 idiomas	6 idiomas
Documentación	Media	Alta	Baja	Alta
Licencia y distribución	Creative Commons Atribution, Gratuita	Apache 2.0, Gratuita	Creative Commons Atribution Gratuita	GNU y Apache 2.0 Gratuita
Repositorio	Github	Google Developers y Github	Github	Github

objetos con los que interactúa el usuario no son Gráficos Vectoriales Redimensionables (SVG), si no Gráficos de Red Portátiles (PNG).

4.1.6. LEGO Mindstorms Educational EV3

Este software está basado en LabView y se diferencia enormemente de las interfaces que se han señalado hasta el momento, ya que implementa un lenguaje de programación gráfica (G) que utiliza iconos, terminales y cables en lugar de texto o bloques tipo rompecabezas como Blockly. Este lenguaje ha tenido una enorme aceptación en la industria, en la ingeniería asistida por computadora para la automatización de procesos.

4.2. Requerimientos Funcionales

Los requerimientos funcionales son una consecuencia de la investigación realizada acerca de las interfaces de programación de mayor aceptación en el mercado y que se ajustan a los objetivos establecidos. Para ello se establecieron aquellos elementos funcionales que deben estar presentes en el desarrollo.

Otras interfaces de programación estudiadas no aparecen en la descripción realizada en la Tabla 3. Debido a la falta de documentación o tipo de licencia. En particular la interfaz de programación EV3 de LEGO, aportó información valiosa respecto al tipo de lenguaje de programación visual que se debe implementar, ya que a diferencia de Blockly o Scratch, utiliza

íconos y eventos en vez de bloques y estructuras. Sin embargo, en la implementación de este proyecto se establece que la interfaz de programación debe contar con un editor de código basado en Blockly, debido a la gran cantidad de desarrollos educativos que han demostrado éxito utilizando este tipo de lenguaje, entre los casos más exitosos se destacan: App Inventor del MIT que permite crear aplicaciones para Android, Code.org que ha logrado enseñar programación básica a millones de estudiantes en todo el mundo, Wonder Workshop que ha trabajado en el área de robótica educativa a través del robot Dot and Dash, Snapp que permite crear prototipos de aplicaciones para iOS o Android y finalmente LEGO con su programa en versión beta llamado Open Roberta, como alternativa para programar muchos de los productos que tienen en el mercado.

4.3. Requerimientos no funcionales

"La funcionalidad se presupone, la calidad satisface (o no)". (Departamento de Informática, 2010)

Los requerimientos no funcionales corresponden a todo lo que el usuario percibe al interactuar con el software. A veces se denominan requisitos de calidad. Y según la norma ISO/IEC 24765 un requerimiento no funcional no describe lo que hará el software sino cómo lo hará. (ISO, 2010)

Algunos ejemplos de requerimientos funcionales son la amigabilidad, el uso de estándares, la reusabilidad o la interfaz gráfica del sistema. Estos requerimientos se comportan como restricciones de los servicios del sistema, pero deben ser negociables.

Se identificaron aquellos requerimientos mínimos no funcionales que debe tener una interfaz gráfica de programación basada en bloques, la descripción de estos se puede apreciar en la Tabla 4.

4.4. Librerías y herramientas para el desarrollo

Una vez planteados los requerimientos básicos funcionales y no funcionales se realizó una investigación de aquellas herramientas que facilitaran el desarrollo e implementación del sistema. La metodología implementada permite incluir nuevos requerimientos en el proceso o refinar cada uno de los desarrollos o incrementos hasta la entrega final.

QT: Es una biblioteca multiplataforma que permite desarrollar aplicaciones con interfaz gráfica de usuario, al igual que programas para línea de comandos o consolas para servidores, es un software libre y de código abierto. QT utiliza el lenguaje de programación C++ de forma nativa, pero puede ser usado en varios otros lenguajes a través de bindings como PyQt para Phyton, Qt Jambi para Java o QtRuby para el novedoso lenguaje de programación Ruby. Actualmente QT se encuentra disponible para diversos sistemas operativos, incluyendo sistemas tipo unix, OS X, Windows o Linux Embebido. (Blanchette & Summerfield, 2006)

Blockly: Como se ha descrito anteriormente, Blockly es una librería que permite desarrollar editores de programación gráfica basados en bloques. Esta librería está escrita en JavaScript y

Tabla 4. Requerimientos No Funcionales del Sistema

1.	Capacidad de arrastrar, conectar, eliminar, copiar y pegar bloques en un espacio determinado.
2.	Posibilidad de multilenguaje: Inglés – Español
3.	Creación de Ejemplos
4.	Mensajes de Ayuda sobre los bloques
5.	Color de bloques no conectados o aislados diferente a los demás
6.	Editor de código texto para programar
7.	Posibilidad de tener una visita guiada
8.	Botones de Verificar, Subir, Guardar, Abrir
9.	Interfaz para la comunicación serial
10.	Visualización de código texto generado por los bloques
11.	Capacidad de escalar el tamaño de los bloques
12.	Generar una versión portable del sistema y un instalador de la misma.
13.	Opciones de manejo archivos

Python, por lo tanto, es posible integrar Blockly a QT a través de los bindings QtQuick para JavaScript y PyQt para Phyton. Esta integración pretende simular la interacción entre un Webkit que permita recuperar y renderizar páginas o aplicaciones web y una librería de creación de objetos con las características de Blockly.

Python: Este lenguaje de programación multiparadigma y multiplataforma, permite desarrollar interfaces de línea de órdenes, es decir gran parte del back-end del sistema sucede en este punto.

JavaScript: Este lenguaje de programación permite la creación de interfaces de usuario y páginas web dinámicas, todo lo que el usuario percibe se desarrollará a través de este leguaje.

HTML: Este es un lenguaje marcado, es decir una forma de codificar documentos, incorporando etiquetas o marcas que contienen información adicional sobre la estructura y presentación. Este estándar permite definir el contenido creado en JavaScript y organizarlo a conveniencia.

Twitter Bootstrap: Un framework que facilita el diseño de sitios y aplicaciones web. Aporta un sinfín de plantillas de diseño con tipografía, botones, formularios, menús de navegación y otros elementos de diseño soportados en el estándar HTML y CSS.

CSS: Es un lenguaje que se usa para definir y crear la presentación del contenido estructurado en HTML, XML o XHTML. CSS permite separar la estructura y la presentación o estilo.

5. DISEÑO PRELIMINAR

En el diseño preliminar pretende trasformar los requerimientos funcionales y no funcionales, en datos y arquitectura de software.

A continuación se presenta cada uno de los diseños modulares del desarrollo.

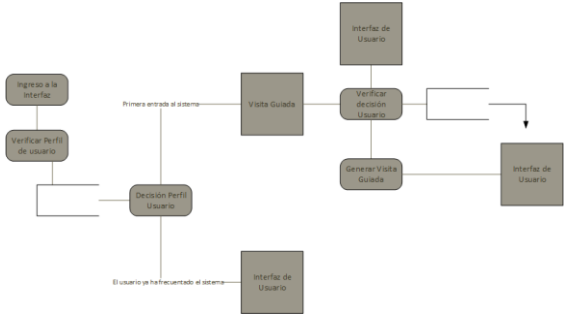


Figura 6. Inicio de la Interfaz

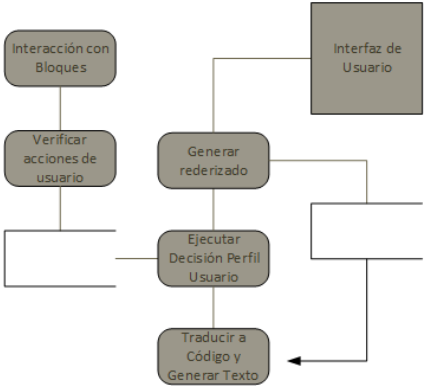


Figura 7. Interacción usuarios y bloques

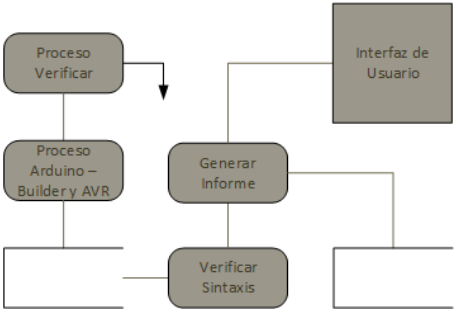


Figura 8. Proceso Verificar Código

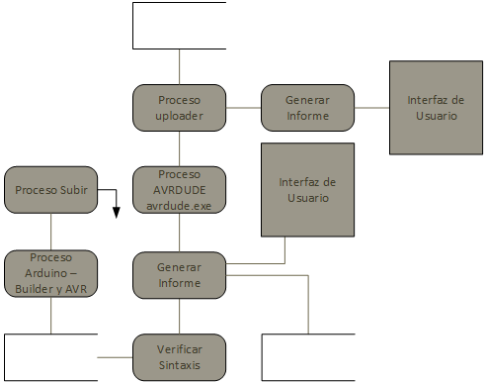


Figura 9. Proceso de Cargar o Subir Programa a la Tarjeta

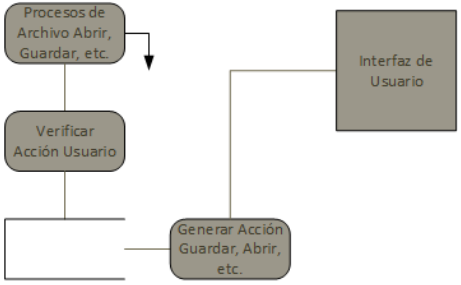


Figura 10. Procesos de Manejo de Archivos

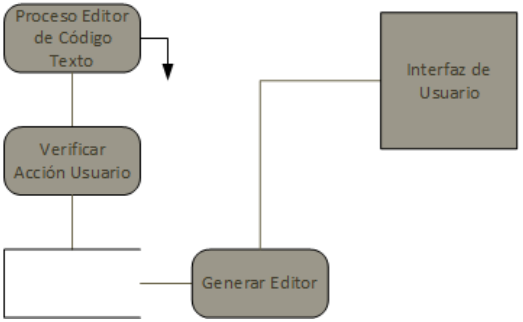


Figura 11. Editor de Código

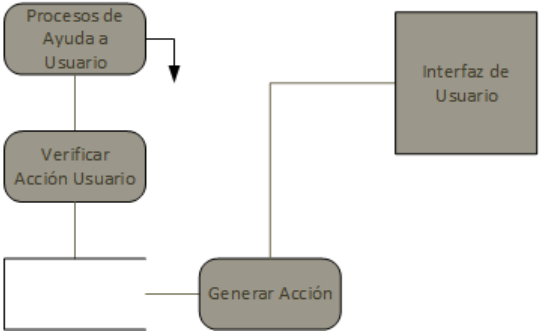


Figura 12. Ayudas

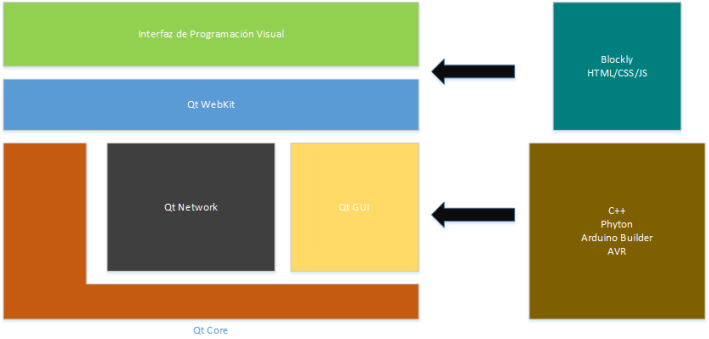


Figura 13. Integración de Desarrollos y Herramientas

5.1. Creación de Repositorio

Este proceso permite generar control sobre el manejo de versiones, documentación y almacenamiento del proyecto, previniendo la perdida de información en el sistema. Para ello se utilizará un repositorio GIT.

Un repositorio GIT es un sistema de control distribuido para el control de versiones, facilitando el trabajo colaborativo. Permite además tener control sobre los cambios de código o archivos del proyecto.

Los comandos que se presentan a continuación, permiten configurar un repositorio:

git init : Comando que permite crear un repositorio nuevo, en el archivo raíz donde se ejecutará el código.

git add <filename> : Este comando agrega aquellos archivos a los cuales se les registrará cambios en el repositorio.

git add. : Agrega todos los archivos para registrar cambios en el repositorio.

git commit –m "commit message" : Para confirmar los cambios de los archivos que se agregaron con el comando **git add**, se requiere realizar un comentario.

git remote add origin <server> : Crea un acceso a un repositorio remoto.

git push origin master : Permite subir cambios al repositorio remoto.

6. DISEÑO DETALLADO DE LA INTERFAZ

En esta sección se describen a detalle las clases y funciones que permiten implementar el sistema:

6.1. Creación de Bloques

Para crear bloques a través del framework Blockly, se debe seguir la siguiente estructura:

```
Blockly.Blocks['analog_write'] = {
 init: function() {
   this.setHelpUrl('');
   this.setColour(Code.colors.azul_dark);
   this.appendDummyInput()
       .appendField(Blockly.Msg.ANALOG_WRITE)
       .appendField(new
Blockly.FieldDropdown([["pin3", "3"] ,
            ["pin5","5"],["pin6", "6"],["pin9", "9"],["pin10",
"10"],
            ["pin11","11"]]),  "pin");
   this.appendValueInput("value")
       .setCheck("Number")
       .appendField(Blockly.Msg.VALUE);
   this.setInputsInline(true);
   this.setPreviousStatement(true);
   this.setNextStatement(true);
   this.setTooltip(Blockly.Msg.TOOLTIP_ANALOGWRITE);
 }
};
```

El resultado de este ejemplo es un bloque de éste tipo:

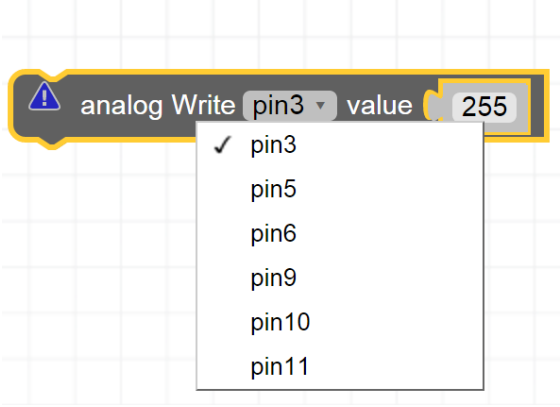


Figura 14. Generación de Bloques

La estructura de creación de bloques comienza con la clase:

```
Blockly.Blocks['key']={
    init: function() {

    }

};
```

Donde **'key'** es una referencia al bloque que se desea construir, esta referencia debe ser única, ya que es un identificador para este objeto en cualquier parte del código.

La función **init()**, permite dar forma al bloque a través de propiedades como **setColour()**, **setHelpUrl()**, **setTooltip()** o **appendDummyInput()**.

6.1.1. Constructor init()

Las propiedades que dan forma al bloque construido son las siguientes:

setHelpUrl (): Permite agregar una url, donde se pueda encontrar información referente al bloque, este campo sirve para documentar el bloque y generar opciones de ayuda al usuario.

setColour(): Esta función establece el color del bloque a construir, el valor de este color es una constante hexadecimal.

appendDummyInput(): Declaración de entrada de campos (fields). Estos campos pueden ser:

- Tipo texto: Texto a visualizar en el interior del bloque.
- Tipo entrada de texto: Se crea un espacio al interior del bloque para que el usuario deba digitar un número o un texto específico.
- Tipo Checkbox: Se crea una pequeña caja de chequeo, internamente funciona como un valor booleano.
- Tipo Dropdown: Permite desplegar diferentes opciones, por ejemplo una serie de pines análogos o digitales, una serie de números determinados, etc.

appendValueInput(key): Esta propiedad permite determinar el tipo de bloque que se quiere generar, ya que, no todos los bloques pueden conectarse entre sí, debe existir una correspondencia, un orden predeterminado.

setCheck(): Set check es una función que recibe un vector con todos los posibles valores que pueden conectarse [“NUMBER”,”Boolean”, “STRING”,”OTHER”]. “OTHER” hace referencia a aquellos bloques personalizados por el usuario.

appendStatementInput(key): Permite conectar cadenas de bloques en el interior del bloque generado. El bloque construido tendrá un comportamiento de contenedor.

Estos bloques difieren de un bloque tipo entrada, porque estos permiten insertar cadenas de bloques y el bloque tipo entrada permite insertar bloques de constantes, funciones o variables.

Un bloque puede ser contenedor y de entrada al mismo tiempo.

Un bloque que se comporte netamente como contenedor se ve visualmente así:

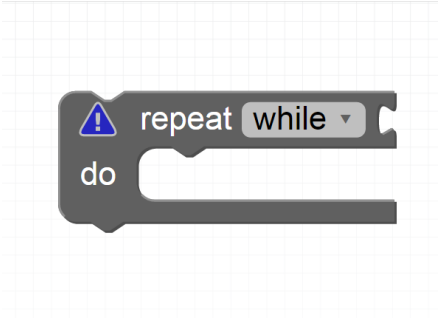


Figura 15. Bloque Contenedor

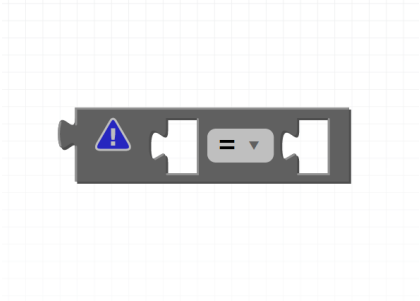


Figura 16. Bloque Entrada

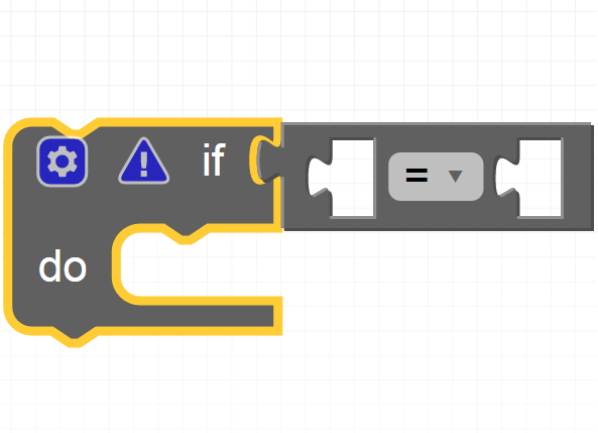


Figura 17. Bloque Contenedor y Entrada

setInputlnline(boolean value): De ser verdadera esta función, las entradas analizadas del bloque construido se presentaran en línea, de lo contrario cada vez que se agregue una entrada, se generará un salto de línea en el interior del bloque.

setTooltip(String msg): Esta función permite desplegar un mensaje de ayuda en el bloque.

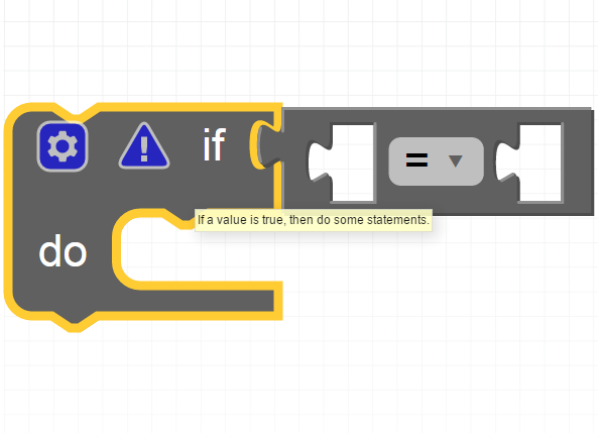


Figura 18. ToolTip

setPreviousStatement(boolean value, String check): En caso de ser verdadero, el bloque tendrá una conexión superior. Se podrá agregar una validación por medio del argumento *check*, para restringir los bloques que se puedan conectar a él.

setNextStatement(boolean value, String check): Si al evaluar la función el valor booleano es verdadero, entonces el bloque poseerá una conexión inferior. Adicionalmente es posible agregar un tipo de salida por medio del argumento check, de esta manera se restringen los bloques que pueden conectarse a él.

setOutput(boolean value, String check): De ser verdadera la evaluación de la función el bloque tendrá una conexión tipo salida. Nuevamente el argumento check permitirá restringir la forma como interactuará el bloque con los demás.

El argumento **check**, se puede declarar de dos formas:

- Vector [“Number”, “Boolean”, “STRING”, “OTHER”].
- String simple “Number”.

En la Figura 19. Se presenta respectivamente una Conexión Superior, Inferior o de Salida:

Figura 19. Tipos de Conexiones

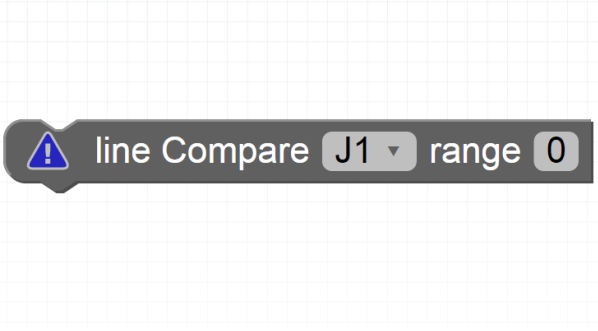


Figura 20. Bloque Conexión Superior e Inferior

appendField(Field, Key): Esta función puede concatenarse con otras para determinar una entrada en el bloque y además permite agregar un campo de dichas entradas. Cada campo debe tener un identificador único o key única, que le permita acceder al valor o estado dinámicamente.

6.2. Funciones heredadas

Una vez generado un bloque, este adquiere ciertas funciones, las cuales permiten manipular el bloque en el área de trabajo o workspace. A continuación, se presentan las más relevantes:

dispose(): A través de esta función se puede eliminar un bloque del workspace.

getConnections_(): Esta función obtiene el un vector que contiene las conexiones del bloque de esta forma es posible acceder a aquellos bloques que se han conectado al bloque en cuestión, además es posible determinar que bloques están desconectados.

getParent(): Retorna el bloque inmediato conectado al bloque en cuestión, si el retorno es nulo (null), existen dos posibilidades:

- El bloque no está conectado a otro bloque
- El bloque se encuentra en la cima de las conexiones, en este caso, es probable que el bloque encierre un conjunto de bloques, es decir que es un bloque contenedor.

getRootblock(): Con esta función se obtiene el bloque que está en la cima de la cadena de bloques, esto quiere decir que la función retorna aquel bloque que contiene al conjunto de bloques donde éste se encuentra conectado, si no hay bloques conectado a él, el resultado, es el bloque mismo.

setParent(newParent): Es posible establecer un "pariente" al bloque, es decir que se le indica el bloque que se quiere conectar a él.

isDeleteable(): Retorna un valor verdadero o falso. Si el valor es verdadero, el bloque se encuentra en un estado donde no puede ser borrado por el usuario.

setDeleteable(boolean value): Si el valor es verdadero, el bloque se encuentra en un estado donde no puede ser borrado por el usuario.

isMovable(): Retorna un valor verdadero o falso. Si el valor es verdadero, el bloque se encuentra en un estado donde no puede ser movido por el usuario.

setMovable(boolean value): Si el valor es verdadero, el bloque se encuentra en un estado donde no puede ser movido por el usuario.

isShadow(): Retorna un valor verdadero o falso. Si el valor es verdadero, el bloque es de tipo sombra, es decir que el bloque nunca podrá ser borrado por el usuario, pero si puede ser superpuesto por otro bloque. Este tipo de bloque se utiliza como conexión a bloques tipo función o sentencia donde se quiere garantizar que existe un bloque conectado.

setShadow(boolean value): Retorna un valor verdadero o falso. Si el valor es verdadero, el bloque adquiere la característica de no poder ser borrado por el usuario.

getColour(): Retorna el color actual del bloque

getFieldValue(key): Con esta función, se obtiene el valor de campo correspondiente al identificador o key. Con ello se pueden obtener los estados y valores de una entrada a un bloque de forma dinámica.

setFieldValue(newValue,key): Permite cambiar los valores o estados de los campos de una entrada a un bloque.

6.3. Parámetros heredados

En la manipulación de los bloques es importante generar parámetros que permitan identificar las características que posee cada bloque, a continuación se presentan las más relevantes:

boolean outputConnection: Cuando este parámetro es verdadero, indica que el bloque tiene una conexión tipo salida.

boolean nextConnection: Si es verdadero, indica que el bloque tiene una conexión inferior.

String type: Retorna el identificador o key del bloque creado, esto permite entender que bloque se está manipulando.

Object workspace: Este parámetro retorna el objeto en el que se encuentran y manipulan los bloques, con este parámetro se tiene acceso a todo el espacio de trabajo generado o workspace.

Array xy_: Retorna a las coordenadas (x, y) de los bloques, de esta forma es posible identificar la posición que tienen en el workspace.

boolean deleteable, movable, editable: Estos parámetros brindan información sobre la situación de un bloque en el workspace. Un bloque puede ser borrado, movido o editado.

Objetc parentBlock: Este parámetro contiene el bloque al que está conectado un bloque determinado.

Array childBlocks_: Es un vector que contiene todos los bloques que están conectados a un bloque determinado.

6.4. *Codificación de Bloques*

Cada bloque generado debe ser traducido o codificado a un lenguaje de programación que pueda ser compilado. Por omisión Blockly permite codificar los bloques a JavaScript, Python, PHP, Dart y XML. En este proyecto se realiza una traducción al lenguaje de programación de alto nivel adaptado desarrollado por Arduino y Wiring implementado en C/C++ pero integrando parte de la sintaxis y estructura de Processing.

Processing: Es un lenguaje de programación de código abierto implementado en Java, que permite crear imágenes, animaciones e interacciones. Es un lenguaje de alto nivel y está enfocado a diseñadores y artistas. (Reas, 2014)

Una de las razones por las que se decide traducir los bloques a este lenguaje es que va permitir interactuar con el Kit de educativo de Robótica Innobot.

Para realizar una traducción de cada uno de los bloques a este lenguaje es necesario utilizar la siguiente estructura:

```
Blockly.Cpp['analog_write'] = function(block) {
 var pin = block.getFieldValue('pin');
 var value = Blockly.Cpp.valueToCode(block,'value',
                                     Blockly.Cpp.ORDER_ATOMI
C);
 var code = 'analogWrite(' + pin + ',' + value + ');\n';
 return code;
};
```

En la estructura anterior se aprecia la creación de un vector llamado “Cpp”, a este vector se le asigna el identificador o key, del bloque que se desea traducir, este identificador debe coincidir con aquel que se asignó en el momento de creación. El vector creado permitirá tener acceso al bloque equivalente por medio del parámetro “block”. De esta forma es posible capturar los campos del bloque, a través de las funciones y parámetros descritos en las secciones 6.1, 6.2 y 6.3.

Una vez capturados los valores necesarios, se realiza una concatenación, que permite generar el código traducido. La variable donde se concatenan los diferentes campos es “code”.

La función que permite obtener el valor de un campo generado es **getFieldValue(‘key’)**.

6.5. *Conexiones Dinámicas*

Las conexiones dinámicas se utilizan para asegurar que ciertos campos tienen bloques conectados o para evitar errores de usuario al intentar conectar un bloque de forma incorrecta o sin seguir la correspondencia lógica, forzando el sistema a desconectar el bloque que intenta utilizar indebidamente. Blockly provee una clase llamada *Connection*, que permite controlar estos eventos.

6.5.1. Parámetros

Object sourceBlock_: Este parámetro permite almacenar el bloque fuente de la conexión, es decir, que la conexión se obtiene a partir del bloque.

Object targetConnection: Retorna la conexión del bloque. Si el bloque está desconectado se retornara un valor nulo (null).

Array check_: Indica el tipo de conexiones que puede tener un bloque, con este parámetro se puede validar que se está conectando.

6.5.2. Funciones

dispose(): Corta todas las conexiones a un bloque en particular.

connect (TargetConnection): Permite enlazar o conectar dos conexiones de bloques diferentes.

disconnect(): Realiza la desconexión de un bloque

targetBlock(): Obtiene el bloque unido a una conexión generada. Sirve para validar que determinado bloque esté unido o enlazado a otro bloque específico.

setCheck(Array check): Establece los tipos de conexiones que se pueden tener.

6.6. Control del Espacio de Trabajo (Workspace)

El espacio de trabajo es el área donde sobre la que se crean los bloques, es dinámico y se puede considerar como un objeto independiente. Para manipular el área de trabajo es importante entender las funciones y parámetros que lo conforman.

6.6.1. Funciones y Parámetros

addChangeListener(Handler): Esta función permite agregar un *listener* al Workspace, lo cual quiere decir, que cada vez que exista un cambio o modificación en el área de trabajo, se ejecuta un evento, dicho evento es capturado por el listener a través del parámetro Handler. Cuando se registra la creación de un bloque o se realiza una conexión o una desconexión, el listener permite que dicha acción pueda ser atendida inmediatamente por el sistema.

getAllBlocks(): Esta función obtiene un vector de todos los bloques que se encuentran en el espacio de trabajo.

newBlock(): Permite crear dinámicamente un bloque en el espacio de trabajo.

getTopBlocks(): Cumple una función similar a getAllBlocks(), pero retorna solo aquellos bloques que no están conectados, o que son contenedores de toda una secuencia de bloques.

6.6.2. Maquetado de la Interfaz

La maquetación hace referencia al proceso de diseño que involucra la organización del espacio para publicar contenido, escrito, visual, audiovisual, en medios impresos o electrónicos.

La razón por la que se realiza el maquetado de la interfaz es que la aplicación desarrollada tiene muchas similitudes con una aplicación web y por lo tanto se puede diseñar utilizando lenguajes marcados como HML y CSS.

Para ello se utiliza un constructor del framework Bootstrap conocido como *Brix.io*.

Brix permite simular el diseño y la funcionalidad en tiempo real, brinda la posibilidad de acceder al código generado, a través de una interfaz sencilla que cuenta con herramientas drag & drop.

Otro constructor que permite el maquetado web a través de las herramientas drag & drop, es *Bootply*.

Bootply integra los frameworks jQuery, AngularJS Bootstrap y muchos más.

6.7. Componentes del diseño de la interfaz a nivel visual

Una interfaz es el canal de comunicación entre el usuario y el sistema. Este canal de comunicación incluye aspectos cognitivos y emocionales. Según Felipe Londoño, "El diseño visual resuelve problemas de comunicación en una forma que va entre la efectividad funcional y el placer estético". (Londoño, 2005)

Aunque este proceso está ligado a la subjetividad, es importante mantener un estándar de diseño sujeto a la simplicidad, reduciendo el diseño a lo esencial, regularizando elementos de diseño a través de formatos y escalas ajustables. (Manovich, 2005)

6.7.1. jQuery

jQuery es una biblioteca de JavaScript, que simplifica la integración a documentos HTML y la generación de animaciones.

La estructura típica de jQuery es la siguiente:

- $(".Nombre_clase").funcion_name()
- $("#Nombre_id").funcion_name()

Siempre iniciará la sentencia con el signo de pesos "$". Luego entre comillas se hace referencia al objeto que será manipulado, esta referencia puede hacerse a través del atributo clase o id usado en HTML; en caso de usar el atributo "clase" se utiliza un ".", o si es por medio del atributo "id" el carácter "#". Finalmente se agrega la función que se desee usar.

A través de jQuery se controlan los eventos de la interfaz cuando el usuario realiza una acción

determinada sobre la interfaz, como por ejemplo un "click".

Una estructura de control de eventos jQuery tiene los siguientes elementos:

$("#name_id").on("name_event",handler);

• name_id: id para determinado objeto de la interfaz, haciendo uso del atributo "id" de HTML.

• funcion "on" : Función usada para asignar un evento al objeto antes declarado.

• name_event: Evento que será enlazado al objeto.

• handler: nombre de la función que se ejecutara cuando ocurra el evento.

Los eventos más usados son:

• **"click":** Ocurre un "click en el objeto".

• **"load":** Este evento se da cuando el documento HTML termina de cargar.

• **"mouseover":** Ocurre cuando el cursor del mouse está sobre el elemento u objeto

• **"mouseout":** Ocurre cuando el cursor sale del elemento u objeto.

• **"mousedown":** Ocurre cuando se sostiene un "click" ya sea derecho o izquierdo sobre el elemento u objeto.

• **"mousemove":** Este evento ocurre cuando el cursor del mouse se mueve mientras está sobre el elemento u objeto.

• **"mouseup":** Ocurre cuando se suelta el "click".

Una alternativa al manejo del evento "click" es a través de Blockly, ya que posee algunas funciones para el manejo de eventos, una de ellas es *bindClick().*

bindClick(): Permite manejar el evento click a través de la siguiente forma: binClick ("name_id", handler);

La diferencia con jQuery reside en que esta función solo admite el atributo "id", dejando de lado un atributo tipo "clase"

La implementación de los eventos verificar y subir se realiza con la siguiente estructura:

```
BlocklyApps.bindClick('verificar',
  function(){
     Code.renderContent();
     try{
        QtFunctions.compilar();
     }catch(err){
        console.warn(err);
     }
  });

BlocklyApps.bindClick('subir',
  function(){
     Code.renderContent();
     try{
       QtFunctions.subir();
     }catch(err){
        console.warn(err);
     }
  });
```

7. COMUNICACIÓN DE LOS DIFERENTES MODULOS DE LA INTERFAZ

La interfaz desarrollada tiene características de una aplicación web, esto quiere decir que en primera instancia no es posible que la aplicación acceda a ciertos recursos de un computador, por si misma.

Es por ello que el desarrollo está estructurado, de tal forma que se crea un directorio que contiene todos los recursos (HTML, JavaScript y CSS). Y otro directorio que contiene el desarrollo del WebKit, generando el enlace entre el sistema y la aplicación web.

Una aplicación web tiene varias ventajas sobre una convencional, en primera instancia la portabilidad en términos de recursos es mucho mayor, la escalabilidad de la interfaz no se está restringida, es posible refinar y actualizar con mayor frecuencia la aplicación, la interfaz visual solo requiere un navegador para ser testeada. Otra de las grandes ventajas, es que está desarrollada a partir de un código que se puede implementar en cualquier sistema operativo y dispositivo. Sin embargo, se debe aclarar que esta interfaz necesita algunos recursos que solo pueden ser generados en un sistema operativo a la vez y que no coinciden con las características de un aplicación web. Este es el caso del compilador arduino-builder y el programa AVRDUDE que permite convertir código ejecutable en un archivo de texto hexadecimal, que es finalmente el código que se carga en la tarjeta Innobot, UNO, Leonardo u en otras con un firmware similar.

QT provee un conjunto de librerías para la conexión de procesos externos a la aplicación a través del acceso al WebKit generado a través de esta herramienta.

7.1. QWebFrame

Es la representación del frame web que se usa en la aplicación desarrollada, con ello se tiene acceso a las diferentes funciones y parámetros de JavaScript implementadas en el framework Blockly. Esta es la comunicación inicial bidireccional web – QT.

La implementación del QWebFrame se consigue a través de las siguientes estructuras:

7.1.1. EvaluateJavaScript()

La función evaluateJavaScript(**cost Qstring** &scriptSource), implementa un script, que retorna el valor de la sentencia ejecutada.

```
QString MainWindow::getCode() {
 QWebFrame *frame = ui->webView-
>page()->mainFrame();
 QVariant xml =frame-
>evaluateJavaScript("Blockly.Cpp.w
orkspaceToCode();");
 return xml.toString();
}
```

Con ello se sustrae el código generado por una secuencia de bloques en el workspace de la aplicación y se almacena en la variable *QVariant*. Dicha variable deberá pasar por un casting a String, lo que permitirá acceder al código que se va a compilar.

Con esta función se obtiene un sentido de comunicación desde Blockly a QT.

7.1.2. addToJavaScriptWindowObject()

A través de esta función, se genera una comunicación desde QT hacia Blockly.

addToJavaScriptWindowObject(**const QString &** name, **QObject** *object)

El parámetro *name* es el identificador del objeto al que se quiere hacer referencia, esto se hace a través de un identificador o key.

El segundo parámetro es la representación del objeto en JavaScript, de esta forma la función puede reconocer las funciones y parámetros de dicho que le pertenecen.

La implementación de esta función es:

```
void MainWindow::addQtcode(){
ui->webView->page()->mainFrame()->

     addToJavaScriptWindowObject("
QtFunctions",this);
}
```

- **mainWindow:** hace referencia a la ventana principal de la aplicación.
- **QtFunctions:** Es el nombre de la variable que amacena el objeto. Es decir que es posible acceder al objeto mainWindow a través de ella.

7.1.3. *Estructuras en QT de las funciones compilar() y subir()*

Estas son funciones que se implementan en QT a través de las siguientes estructuras:

```
BlocklyApps.bindClick('verificar',
    function(){
        Code.renderContent();
        try{

QtFunctions.compilar();
        }catch(err){
            console.warn(err);
        }
    });
BlocklyApps.bindClick('subir',
    function(){
        Code.renderContent();
        try{
            QtFunctions.subir();
        }catch(err){
            console.warn(err);
        }
    });
```

7.1.4. *Proceso de compilación y carga o subida del código generado*

Para este proceso se crea un archivo temporal donde se almacenará el código capturado.

```
QFile tmpFile(settings-
>tmpFileName());
    if (tmpFile.exists()) {
        tmpFile.remove();
    }


tmpFile.open(QIODevice::WriteOnly);
```

Se obtiene el código desde el workspace:

```
QString code = this->getCode();
```

Se escribe el código en el archivo temporal:

```
tmpFile.write(code.toLocal8Bit());
tmpFile.close();
```

Se crea un parámetro tipo String y se relaciona con el objeto QProcess, concatenando la acción que se desea realizar, es decir, si se desea compilar el código se agrega "--verify" y si se desea subir el código a la tarjeta entonces se utiliza "--upload".

```
QStringList arguments;
arguments << "--verify";
```

Luego se debe concatenar el tipo tarjeta y el número del puerto. Ambos parámetros son variables y deben ser capturados.

```
arguments << "--board" << "Innobot"
arguments << "--port" << "com3"
```

De igual forma se concatena el contenido del archivo temporal. Es decir el código generado.

```
arguments << settings->tmpFileName();
```

Finalmente, se inicia el proceso mediante la función *start* de Qprocess.

```
QProcess *process;
process->start(
"InnobotC/InnobotC_debug.exe" ,
arguments);
```

El primer argumento que recibe la función start es la ruta donde se encuentra el .exe que hace la compilación y carga backend del código generado a tarjetas con arquitectura AVR como el Innobot, Arduino UNO o Arduino Leonardo.

El InnobotC es una adaptación de Pygmalion Robotics® del Arduino IDE, compatible con el Kit de Robótica Educativo Innobot.

8. COMPILACIÓN A WINDOWS Y LINUX

QT es una aplicación multiplataforma, que permite construir proyectos para Linux, Windows y OS X. La generación de un ejecutable de la aplicación se puede realizar siempre y cuando se instale correctamente el compilador de QT para la plataforma deseada.

- Para Windows: QT utiliza el compilador MinGW o Visual Studio.
- Linux: Compilador GCC.

Dependiendo de la plataforma en la que se desee compilar se debe seleccionar en la ruta indicada el compilador a utilizar:

Ruta: QT >> Tools >> Options >> Build&Run >>Compilers

Una vez seleccionado el compilador, se procede a compilar de nuevo el proyecto.

9. RESULTADOS

La investigación realizada y el desarrollo final es consecuencia de los requerimientos funcionales y no funcionales identificados. A continuación, se presentan los resultados obtenidos tras cada incremento o entrega generada, a través de la implementación de la metodología de desarrollo de software propuesta.

9.1. Primera Entrega

En la primera entrega se implementó el framework Blockly para generar bloques con las características propias de la librería Innobot. Estos bloques se pueden conectar, eliminar, copiar y pegar en el espacio de trabajo o workspace, además cada bloque se logra traducir a lenguaje de programación escrito.

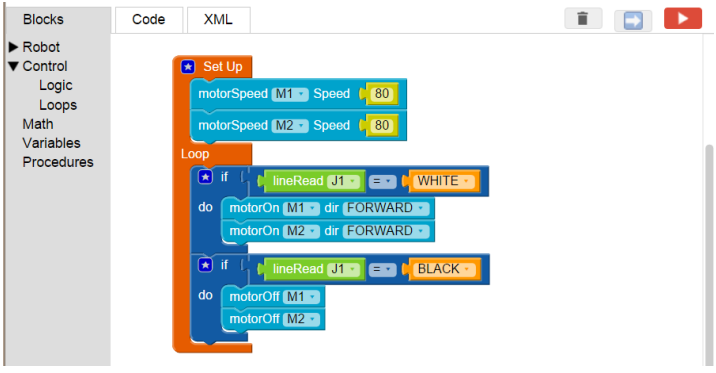


Figura 21. Primera entrega de la Interfaz

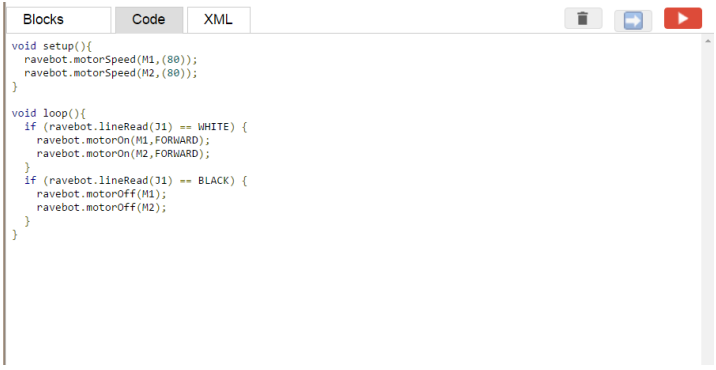


Figura 22. Traducción a código Primera Entrega

9.2. *Segunda Entrega*

Se diseñó una nueva interfaz propia a través del framework Bootstrap y las diferentes herramientas de maquetación. Se cambia el sentido del Toolbox, es decir el objeto donde se despliegan los bloques que el usuario escoge a través de categorías. Finalmente, se agrega la posibilidad de cambio de idioma español – inglés.

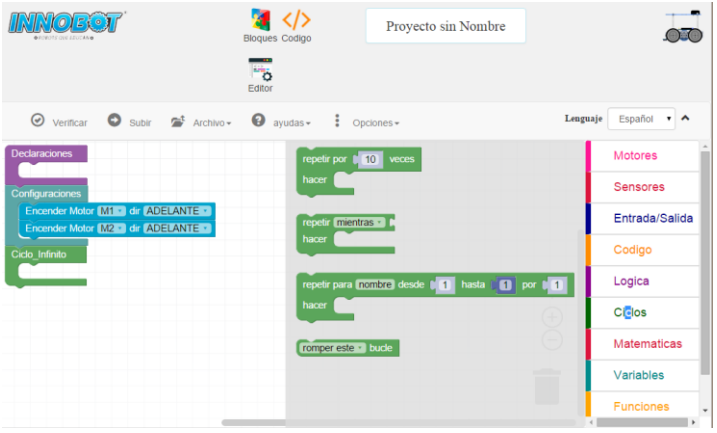


Figura 23. Segunda entrega de la Interfaz

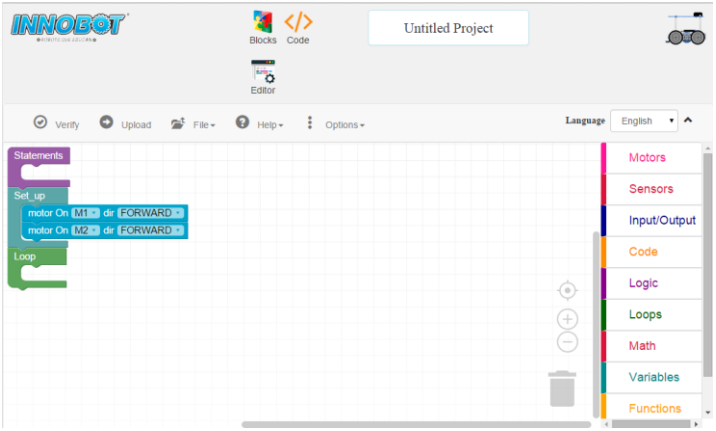


Figura 24. Segunda entrega de la Interfaz

9.3. *Tercera Entrega:*

En la tercera entrega se incluye un editor de código implementado en JavaScript. Adicionalmente, se implementa el framework de QT para extender la comunicación de la interfaz diseñada a recursos externos como InnobotC, a través de un WebSocket.

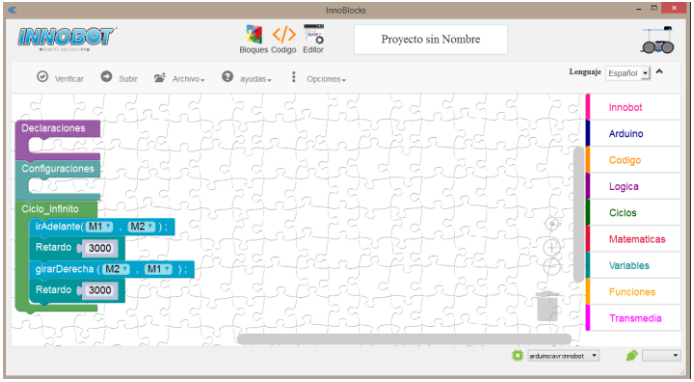


Figura 25. Tercera entrega de la Interfaz

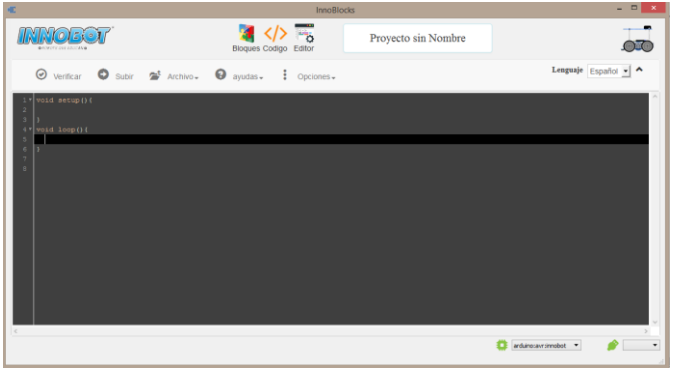


Figura 26. Tercera entrega de la Interfaz

9.4. *Cuarta Entrega:*

Se implementa todo el sistema, implementado las funcionalidades para compilar y subir programa a la tarjeta capturando el código generado por los bloques presentes en el espacio de trabajo, además, se ajusta la interfaz gráfica, generando nuevos bloques con un lenguaje más humano. Los bloques que están desconectados cambian de color.

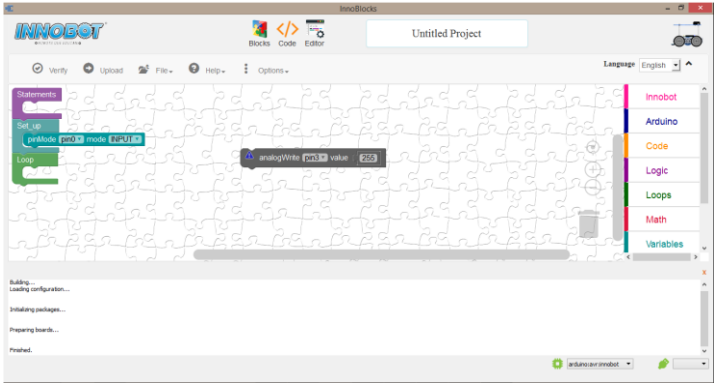


Figura 27. Cuarta entrega de la Interfaz

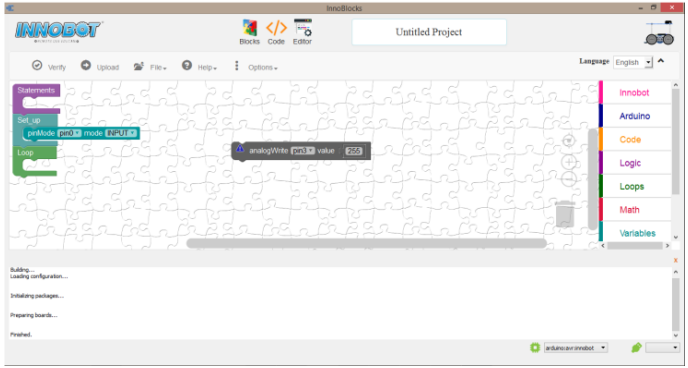


Figura 28. Compilación de la tarjeta Innobot

9.5. *Quinta Entrega:*

En la última entrega se realizó un ajuste gráfico de la interfaz y de los bloques, se generaron los diferentes ejemplos, se agregaron sonidos, objetos y funcionalidades de interacción con el usuario. Se optó por una interfaz más sencilla e intuitiva.

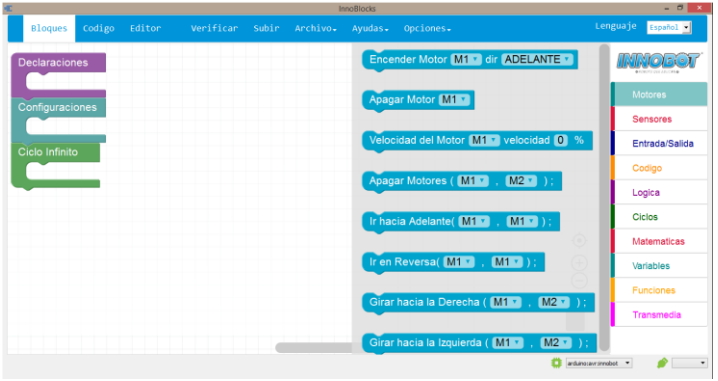
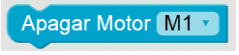


Figura 29. Quinta entrega de la Interfaz

9.6. *Bloques de la librería Innobot y Arduino*

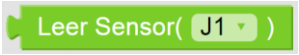
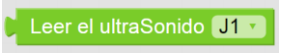
La librería Innobot maneja funciones predeterminadas para el control de sensores y motores, utilizando un lenguaje ideal para aquellas personas que están comenzando a programar. Los bloques que pertenecen a esta librería son:

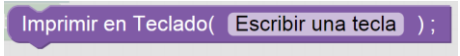
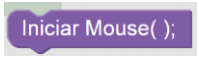
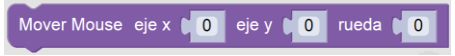
Tabla 5. Bloques para el control de motores

Función	***Representación en Bloque***
motorOn(MOTOR, DIRECCIÓN);	
motorOff(MOTOR);	

motorSpeed(MOTOR, PORCENTAJE);	
turnRight(MOTOR1,MOTOR2);	
turnLeft(MOTOR1,MOTOR2);	
goForward(MOTOR1,MOTOR2);	
goReverse(MOTOR1,MOTOR2);	
motorsOff(MOTOR1,MOTOR2);	

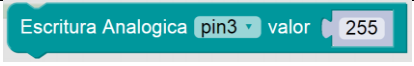
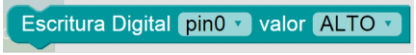
Tabla 6. Bloques para el manejo de sensores y periféricos

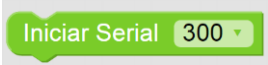
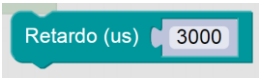
Función	*Representación en Bloque*
sensorRead(SENSOR);	
lineSensor(SENSOR);	
ultrasoundRead(SENSOR);	
lineCompare(SENSOR, Rango);	

Keyboard.begin();	
Keyboard.print(CHAR);	
Mouse.begin();	
Mouse.move();	

No solo se diseñaron los bloques de las diferentes funciones de la librería Innobot, si no que teniendo en cuenta que el lenguaje de programación implementado es la adaptación de Processing realizada por Arduino, se deben crear bloques que representan las funciones principales, como *digitalWrite*, *pinMode*, entre otros.

Tabla 7. Bloques Principales de Arduino

Función	*Representación en Bloque*
analogWrite(pin,nivel);	
digitalWrite(pin, HIGH-LOW);	
Serial.print();	

Serial.begin();	
analogRead(pin);	
digitalRead(pin);	
pinMode(pin, MODO);	
tone(pin, frecuencia);	
tone(pin);	
pulseIn(pin, ENTRADA);	
delay(milisegundos);	

10. POTENCIAL

La educación tiene retos muy importantes, el primer reto es la construcción de valores, para luchar por la construcción de una mejor sociedad. Las realidades políticas, sociales, ideológicas, económicas y ambientales, pueden ser un impedimento para la realización personal y comunitaria. La constitución del ser para construir saber a través de la esperanza y el autoconocimiento. La valoración de la cultura para generar sentido de pertenencia. La educación debe promover la motivación de los estudiantes a asumir responsabilidades sociales, que los impulse a cambiar realidades a través del conocimiento.

Pero el acceso al conocimiento, a la información, también es gran impedimento para el desarrollo intelectual, es por esto que la implementación de herramientas para generar ambientes de aprendizaje accesibles, modificables, de uso abierto y sin fines comerciales, pueden ser fundamentales en la gran revolución educativa que se ha estado generando en países como Finlandia o Estados Unidos.

La interfaz que se ha diseñado no solo soluciona problemas de accesibilidad, sino que está orientada a la resolución de problemas a través de la lógica, la tecnología y el pensamiento computacional. El componente de robótica educativa a través de la competencia promueve la resiliencia de los estudiantes, es decir la capacidad de superar situaciones adversas como la frustración y canalizar las adversidades hacia la auto superación y la búsqueda de nuevos conocimientos. En conjunto se ha contribuido a la creación de un ambiente de aprendizaje que promueve los valores y el amor a la luz del humanismo cristiano, donde primero se entiende que el ser está inmerso en un conjunto de situaciones que pueden atentar contra su desarrollo normal, pero que con las herramientas, maestros e información adecuada y suficiente pueden ser generadores de esperanza para la sociedad.

11. CONCLUSIONES

1. Una vez analizada la información disponible en diversas fuentes para conocer el estado del arte, se concluye que existen diversas interfaces de programación que intentan desarrollar competencias en ciencia y tecnología. Los estudiantes que acceden a estas herramientas ejercitan la lógica y el pensamiento computacional, se motivan a la resolución de problemas y al uso consciente de las TIC.

2. La tendencia STEAM, es acertada, porque el crecimiento acelerado de la población genera nuevos retos, serios y grandes problemas que deben ser resueltos para asegurar la supervivencia y la calidad de vida de la sociedad. Esto solo es posible si más jóvenes se interesan en estas áreas del conocimiento, integrando una educación basada en valores sociales, sentido de pertenencia e investigación.

3. El constructivismo y el construccionismo son metodologías educativas que se ajustan a las necesidades sociales del mundo actual, en este trabajo se concluye, que la experimentación, la indagación, la confrontación, el debate son actividades que deben ser recurrentes en cualquier proceso de aprendizaje, donde el maestro debe ser una guía hacia el insondable conocimiento. El constructivismo tiene como objetivo el hacer para aprender y eso es precisamente lo que se ha logrado con la construcción de la interfaz de programación, ya que el usuario debe afrontar un ambiente de aprendizaje donde está invitado construir, investigar, utilizar la lógica y su propio bagaje de conocimientos, para resolver paulatinamente problemas de mayor complejidad.

4. La metodología implementada para el desarrollo del sistema, permitió que el sistema se construyera en el tiempo propuesto para la implementación, ya que propone un método eficaz donde se divide el trabajo en pequeñas partes y se entrega regularmente un avance. Este método es muy flexible, puesto que los requerimientos se pueden ajustar conforme a la aceptación del usuario o los diferentes stakeholders. La eficiencia está marcada por la experiencia del desarrollador y los conocimientos generales y específicos. Para este trabajo se contó con la fortuna de conocer diferentes frameworks y la asesoría de Pygmalion Robotics® en las diferentes etapas de desarrollo.

5. El diseño preliminar permitió entender los diferentes módulos, procesos y procedimientos a implementar, con ello se identificaron las funciones principales y se determinó la mejor forma de proceder según el caso. Este procedimiento es complejo, pero muy útil en la etapa de codificación.

6. En el diseño detallado, se crearon los diferentes módulos del sistema, para ello se desarrolló toda la parte gráfica a través de frameworks y lenguajes de programación marcados como HTML y orientados a objetos como JavaScript. Este módulo, se conecta al módulo del WebKit, donde se logra extender el uso de recursos de la aplicación hacia el sistema donde se va a ejecutar, para ello se contó con QT. Finalmente se conecta el

módulo de compilación y carga de código generado a la tarjeta Innobot, para ello se implementó un proceso de ejecución de subprogramas como avrdude y arduino-builder.

7. Una lista de control del sistema permitió administrar el desarrollo de la aplicación con mayor eficiencia, ya que incluye un historial de las versiones generadas, la lista de funcionalidades y tareas a implementar y la definición de funciones principales.

8. Los cinco incrementos generados, son una muestra clara de la evolución de la interfaz hasta el producto final escalable, el sistema presenta un gran margen de refinamiento, pero la versión final es estable y cumple con los diferentes requerimientos. La validación del sistema realizada por Pygmalion Robotics® fue satisfactoria tras cada entrega.

9. Se logró compilar la aplicación tanto para el sistema operativo Windows como para Linux, esto permitió que la interfaz tuviera una mayor aceptación y accesibilidad.

10. La implementación del sistema de comunicación y trasferencia de datos entre el software y la unidad de control del robot Innobot fue exitosa y en conjunto se convierten en un ambiente de aprendizaje que integrado a una metodología educativa adecuada y a un maestro dispuesto e interesado por la tendencia STEAM, puede contribuir al desarrollo intelectual de nuevas mentes interesadas en áreas de ciencia y tecnología.

11. El éxito de las empresas que desarrollan software y aplicaciones tecnológicas, reside en la adaptación rápida al mercado y a la identificación de las necesidades en el mismo, pero bajo las características actuales del mundo y el crecimiento acelerado de la información es muy difícil determinar si una aplicación será exitosa o no. Esto se mitiga a través de los test de usuario y la implementación de metodologías agiles basadas en modelos tradicionales como el incremental e iterativo. La combinación entre el método, la experiencia y el seguimiento de unas buenas prácticas de desarrollo aseguran que el nivel de competitividad de una empresa sea mucho mayor, puesto que comete menos errores al entrar al mercado y tiene una ventaja temporal considerable respecto a otros.

12. Las aplicaciones web para escritorio, tienen muchas ventajas frente al software tradicional, se resalta, la portabilidad, la escalabilidad, la capacidad de correr en diversas plataformas sin consumir muchos recursos, la implementación final es liviana, es posible liberar una versión web y otra escritorio, dependiendo de las características del proyecto, sin necesidad de modificar el diseño a nivel visual. Sin embargo resulta irrisorio que el software de escritorio convencional desaparezca, puesto que estos desarrollos son más robustos, mucho más confiables y privados, estas aplicaciones permiten desarrollar procesos más complejos y pueden acceder a los recursos del sistema sin tantas limitaciones.

13. Se observó una tendencia clara respecto a las demás interfaces de programación y es que cualquier empresa, entidad u organización que pretenda desarrollar el talento humano a través de la ciencia y la tecnología utilizando herramientas concretas como la robótica educativa, debe presentar entre sus productos y posibilidades de aprendizaje un desarrollo con características similares a la interfaz diseñada. Ya no como ventaja competitiva, si no, como una necesidad para seguir evolucionando en el mercado.

14. Los desarrollos implementados bajo la licencias GNU o Apache 2.0 permiten crear una comunidad de desarrolladores e interesados que pueden aportar al crecimiento de la aplicación, colaborativamente y en muchas ocasiones en menor tiempo del que una sola organización podría hacerlo.

15. Un sistema de control de versiones integrado a un repositorio soluciona muchos problemas en el proceso de desarrollo, siendo particularmente útil en la depuración de errores y el refinamiento de la aplicación. Además se resguarda el proyecto frente a la perdida de información por diversas causas.

REFERENCIAS

Arceo Barriga, F. D., & Hernández Rojas, G. (2004). *Estrategias Docentes para un Aprendizaje Significativo.* Ciudad de Mexico: Universidad Nacional Autónoma de México.

B. Gonzalez, H., & J. Kuenzi, J. (2012). Science, Technology, Engineering, and Mathematics (STEM) Education: A Primer.

B. Gonzalez, H., & J. Kuenzi, J. (2012). Science, Technology, Engineering, and Mathematics (STEM) Education: A Primer.

Betancur, M. J. (2013). *UPB_autoArt_guía_Word.* Recuperado el 07 de agosto de 2013, de UPB_AutoArt_plantilla_Word: http://kosmos.upb.edu.co/web/uploads/articulos/(A)_UPB_AutoArt_962.zip

Betancur, M. J. (2013b). *Instructivo General UPB_autoArt.* Recuperado el 03 de marzo de 2013, de http://kosmos.upb.edu.co/web/uploads/articulos/(A)_UPB_AutoArt_962.zip

Betancur, M. J., Moreno, J. A., Moreno-Andrade, I., & Buitrón, G. (2006). Practical optimal control of fed-batch bioreactors for the wastewater treatment. *Int. Journal of Robust and Nonlinear Control, Special Issue on Control of (Bio)Chemical Reacting Systems, 16*, 173-190.

Blanchette, J., & Summerfield, M. (2006). *C++ GUI Programming with Qt 4.* Prentice Hall.

Carretero, M. (1997). *La construcción de la memoria Histórica.* Buenos Aires: Paidós.

Departamento de Informática. (2010). *Ingeniería del Software.* Madrid: Universidad Carlos III de Madrid.

Folmer, E., & Bosch, J. (2004). Architecting for usability: a survey. *Journal of systems and software*, 61-78.

Gonzalez B., H., & Kuenzi J., J. (2012). Science, Technology, Engineering, and Mathematics (STEM) Education: A primer. *Congressional Research Service*, 2-9.

Google. (28 de 05 de 2015). *Google Developers.* Obtenido de Google Developers Web Site: https://developers.google.com/blockly/

Green, D., & DiCaterino, A. (1998). A Survey of System Development Process Models. *Center for Technology in Government*, 3-6.

Gudmundsdottir, S., & Shulman, L. (1987). *Pedagogical content knowledge in social studies.* Scandinavian Journal of Educational Research.

Guynn, J. (26 de Febrero de 2013). Silicon Valley launches campaign to get kids to code. *Los Angeles Times.*

Hassan, Y., Fernández, F. J., & Iazza, G. (2004). Diseño web centrado en el usuario: usabilidad y arquitectura de la información. *Hipertext.net*, 1-14.

Husén, T., & Postlethwaite, T. N. (1989). *The International Encyclopedia of Education.* Chicago: Pergamon Press.

INTECO. (2009). *Ingeniería del software: Metodologías y Ciclos de Vida.* Ministerio de industria, turismo y comercio de España.

International Organization for Standardization. (1998). Guidance on Usability. *ISO 9241-11*.

ISO. (2010). *Systems and Software Engineering ISO/IEC/IEEE 24765.* ISO.

Londoño, F. C. (2005). *Interfaces de las Comunidades Virtuales.* Manizales: Uiversidad de Caldas.

M. Wing, J. (2006). *Carnegie Mellon University.* Recuperado el 10 de Junio de 2015, de Computational Thinking: https://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf

Manovich, L. (2005). *El lenguaje de los nuevos medios.* Paidos Iberica.

Milonovich, B. (2013). *Scratch Cookbook.* Packt Publishing.

Ministerio de Educación Nacional. (2005). *Conpes 3360.* Bogotá D.C.

Moreno, M., Chan, M., Pérez, M., Ortiz, M., & Viesca, A. (1998). *Desarrollo de ambientes de aprendizaje en educación a distancia. Textos del VI Encuentro Internacional de Educación a Distancia.* Guadalajara: Universidad de Guadalajara.

Nacional, M. d. (2005). *Conpes 3360.* Bogotá D.C.

Nielsen, J. (1993). *Usability Engineering.* Mountain View: Academic Press.

Papert, S. (1986). *Constructionism: A New Opportunity for Elementary Science Education.* Massachisetts: Massachusetts Institute of Technology.

Perelman, L. J. (1992). *School's Out: Hyperlearning, the New Technology, and the End of Education.* San Jose State Univ.: William Morrow & Co.

Reas, C. F. (2014). *Processing A Programming Handbook for Visual Designers and Artists.* MIT.

RutaN. (2014). Programa Horizontes: Nuestra herencia es conquistar fronteras. *Programa Horizontes*, 1-10.

Sanders, B. (2009). STEM, STEM Education, STEMmania. Obtenido de http://esdstem.pbworks.com/f/TTT%2BSTEM%2BArticle_1.pdf

Sanders, B. (2009). STEM, STEM Education, STEMmania.

Sommerville, I. (2005). *Ingeniería del software* (Séptima ed.). España: Pearson.

Toffler, A. (1990). *Future Shock.* Bantam Books.

Tsupros, N., Koher, R., & Hallinen, J. (2009). STEM education: A project to identify the missing components, Intermediate Unit 1 and Carnegie Mellon, Pennsylvania.

UNESCO. (07 de Junio de 2010). *UNESCO.* Obtenido de UNESCO: http://www.unesco.org/new/es/media-services/single-view/news/engineer_shortage_a_threat_to_development_underlines_unescos_first_global_report_on_engineering/

Vásquez Giraldo, A. L. (2014). *Hacia un perfil docente para el desarrollo del pensamiento computacional basado en la educación STEM para la media técnica en desarrollo de software.* Medellín.

AUTOR

Daniel Andrés HERNÁNDEZ LONDOÑO. Egresado próximo a graduarse del programa de Ingeniería Electrónica. Líder de Investigación y Desarrollo en Pygmalion Robotics®, facilitador del programa Innobótica durante el año 2014, entrenador de robótica en SRI 2014 y 2015, expositor en Bogotá Robótica 2015 y Una Semana para TIC en Medellín.

ANEXO 1

Tabla 8 - Lista de Control del Sistema

Primera Entrega			
Funcionalidades	**Ubicación**	**Archivos de Implementación**	**Nivel de Refinamiento**
Generación de Bloques de la librería Arduino – Innobot Traducción de Bloques	../html/javascript	pygmablocks.js	Se necesitan crear nuevas funcionalidades en el control de bloques. Se deben reorganizar los bloques en el Toolbox y eliminar los bloques que no son de interés.
	../html/javascript	customInnoblock.js	
	../html/javascript	build.py	
	../html/javascript/core/	blocks.js	
	../html/javascript/core/	workspace.js	
	../html/javascript/core/	toolbox.js	
Segunda Entrega			
Funcionalidades	**Ubicación**	**Archivos de Implementación**	**Nivel de Refinamiento**
Diseño de la Interfaz a nivel visual y Cambio de sentido del Toolbox.	../html/javascript	pygmablocks.js	La interfaz gráfica tiene mucho margen de mejora, se desea adicionar sonidos y utilizar nuevos diseños hasta encontrar el más óptimo, customInnoblock.js contiene funcionalidades de los botones. template.soy e index.html la maquetación.
	../html/javascript	customInnoblock.js	
	../html/javascript	build.py	
	../html	index.html	
	../html/javascript/core/	workspace.js	
	../html/javascript/core/	toolbox.js	
	../html	template.soy	
	../html/javascript/lenguajesTemplate/	en.js y es.js	
Tercera Entrega			
Funcionalidades	**Ubicación**	**Archivos de Implementación**	**Nivel de Refinamiento**
Editor de Código, Framework QT y Web Socket	../src/	mainwindow.h, mainwindow.ui y mainwindow.cpp	Con estos archivos se configura el proyecto de QT y se desarrolla el WebSocket, los archivos de mainwindow generan la conexión entre Blockly y el sistema. El editor se implementa en index.html
	../src/	main.cpp	
	../src/	jswebhelpers.h y jswebhelpers.cpp	
	../src/	src.pro	
	../html/	index.html	
	../html/	template.soy	

			y template.soy.
Cuarta Entrega			
Funcionalidades	**Ubicación**	**Archivos de Implementación**	**Nivel de Refinamiento**
Editor de Código, Framework QT y Web Socket	../src/	mainwindow.h, mainwindow.ui y mainwindow.cpp	Con estos archivos se configuran las rutas de acceso a los recursos del sistema como InnobotIDE. Se realiza un refinamiento general sobre los archivos principales de la primera entrega.
	../src/	main.cpp	
	../src/	jswebhelpers.h y jswebhelpers.cpp	
	../src/	src.pro	
	../src/	graphwidget.h y graphwidget.cpp	
	../src/	settingsstore.h y settingsstore.cpp	
Quinta Entrega			
Funcionalidades	**Ubicación**	**Archivos de Implementación**	**Nivel de Refinamiento**
Implementación de todo el sistema y Depuraciones.	../src/ ../html/	mainwindow.h, mainwindow.ui mainwindow.cpp pygmablocks.js customInnoblock.js build.py index.html workspace.js toolbox.js template.soy	Se prueban todos los componentes integrados repetidamente y se realizan ajustes sobre cada uno de los archivos principales, agregando nuevas funcionalidades y corrigiendo bugs.

ANEXO 2

QT es una aplicación multiplataforma, que facilita la construcción de aplicaciones de escritorio en plataformas como Linux, Windows y MAC.
Para generar un ejecutable de la aplicación, QT automáticamente detecta el tipo de compilador que debe usar dependiendo del tipo de plataforma instalada.

- Windows : Usa el compilador MinGW o VS (Visual Studio)
- Linux: Usa el compilador GCC.

Si en algún momento es necesario agregar o cambiar el tipo de compilador se pueden seguir los siguientes pasos:

Qt -> tools -> options -> Build&Run -> Compilers

Se muestra una venta como la siguiente:

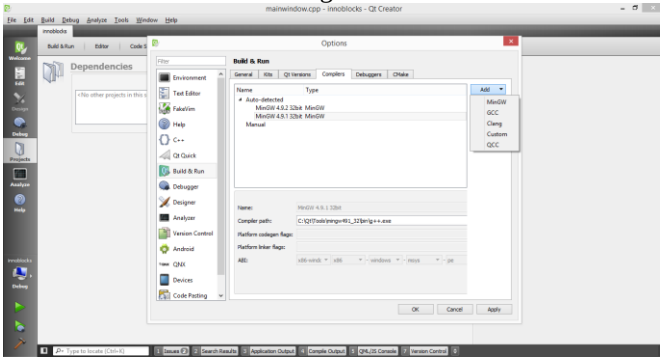


Figura 30. Compilador QT

Presionando en el botón “Add” se pueden encontrar los diferentes compiladores. Finalmente, se posiciona el cursor en la opción marcada con rojo para seleccionar *release*.

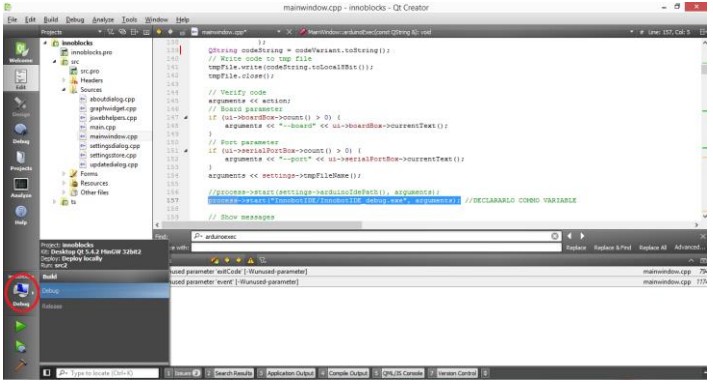


Figura 31. Compilador QT

Es posible cambiar la ruta del directorio donde creará la carpeta, de modo que sea más fácil de ubicar.

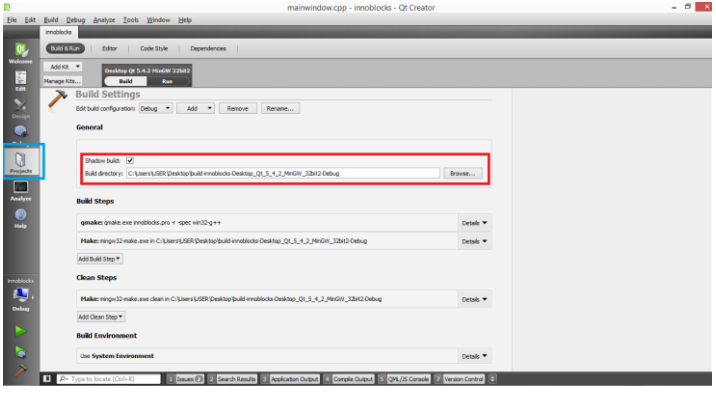


Figura 32. Compilador QT

En este directorio se encuentra el ejecutable, compatible con Windows o Linux, dependiendo del compilador elegido.