



Universidad de Mendoza

Facultad de Ingeniería

Ingeniería en Informática

“Módulo de comunicación en tiempo real entre la aplicación
Alice y un robot creado en Arduino”

Autor: Jonathan, Antolini

Tutor: Dra. Ing. Cristina, Parraga

Fecha: 27 de mayo del 2016

Lugar: Ciudad de Mendoza

AGRADECIMIENTOS

El cierre de la etapa universitaria es uno de los momentos más esperados para cualquier persona que alguna vez emprendió un estudio de educación superior.

Sin dudas no hubiese sido posible sin el apoyo incondicional de mi acotada familia, la cual mi mamá era el único sustento económico. Por esas cosas que nunca entenderé de la vida, ella no pudo verme finalizar pero estoy convencido que estaría más que orgullosa de mí. Este título es para vos.

Gracias también a mi abuela Ana y a mis hermanos Mariano, Nicolás y Diego, el resto de mi familia que siempre que vuelvo al sur (Rio Negro) me esperan contentos, salvando las distancias y desencuentros que muchas veces tenemos, son muy importantes para mí.

A mis amigos del sur, los de las infinitas visitas a Mendoza, y a mis nuevos amigos que obtuve en esta provincia. Si tendría que nombrarlos sería una lista que me daría miedo de olvidarme de algunos. Es por esto que solo nombraré a algunos casos que fueron incondicionales en momentos duros. A Ezequiel que me convenció de venirme a estudiar a Mendoza, sin dudas una muy buena decisión, motivada por el en parte. A Francisco, que compartimos departamentos los primeros años, aprendí mucho sobre como vivir solo con el, a Juan que me abrió las puertas de casa cuando no tenía lugar donde vivir. A Lula, que estuvo las 24 horas cuando más frágil estuve, cuando no había certezas, ella era una luz entre tanta confusión, al pipo y a la Nati, dos hermanos que dejaron la comodidad de vivir solos para agregarme una cama en su departamento. A Agustín y Jimena, mis últimos compañeros de estudios, compañeros de mates y exposiciones en pizarrones, que divertido fue estudiar algunas materias con ustedes.

A la familia López que me apoyó mucho, Cristina con sus llamados antes de rendir.

A la peti, Pau y Lau que me guardaron mis muebles en sus casas, además de sus charlas, mates y abrazos, unas genias. A Pablo Manterola que también me dio hospedaje.

Jonathan, Antolini

Módulo de comunicación en tiempo real entre la aplicación Alice y un robot creado en Arduino

Los chicos de la facu, como le decimos al grupo “los mostros facu” que siempre estuvieron a la hora de estudiar, hacer trabajos, el tener compañeros incondicionales es una condición necesaria para lograr los objetivos, creo que todos los de mi curso, estamos a punto de terminar y eso habla de que fuimos constantes a la hora de estudiar.

Al Mendosur, ese grupo de Mendocinos y Sureños. A la placita, otro grupo de mis compañeros de laburo.

A Fernando Pincirolí, mi primer jefe que siempre estuvo firme a la hora de rendir, a la hora de cursar en horario laboral y para todo lo que un ser humano que trabaja y estudia padece, el siempre incondicional y jamás me olvidare, espero algún día ser un jefe con esa calidez humana como lo es él.

Por último, pero no menos importante a la Universidad de Mendoza, la cual a través de Cristina Parraga me brindaron una beca en las cuotas cuando no podía abonar mi mensualidad. Sin este aporte se me hubiese hecho muy complicado, siempre mi corazón perteneciera a esta institución y espero poder retribuirle a través de docencia al menos un poquito de lo que ellos me dieron.

Y al resto que me vio transitar, sepan que siempre vienen conmigo en este nuevo camino empinado, el de la carrera profesional.

Sin dudas me llevo los mejores recuerdos de todas estas personas, muchas gracias por acompañarme, sin ninguna duda la época cuando fui estudiante fue de las experiencias más lindas de mi vida.

RESUMEN

Este trabajo final titulado Ana busca motivar las vocaciones tecnológicas en los alumnos que están terminando su ciclo académico en el colegio secundario.

La idea es demostrar que a través de un software educativo y un robot que la informática no es cosa aburrida, si no por el contrario, es un mundo apasionante.

El proyecto Ana es un módulo de comunicación en tiempo real entre un software llamado Alice y un auto robótico, buscando así demostrar que con un poco de conocimiento y entusiasmo se puede construir lo que uno se proponga.

[RESUMEN](#)

[INTRODUCCIÓN](#)

[MARCO TEÓRICO](#)

[CAPÍTULO I](#)

[ASIGNATURAS INVOLUCRADAS](#)

[CAPÍTULO II](#)

[ALICE](#)

[Introducción](#)

[Objetivo de Alice](#)

[TICS](#)

[CAPÍTULO III](#)

[PROGRAMACIÓN ORIENTADA A OBJETOS](#)

[Programación en clases y objetos](#)

[Origen](#)

[Conceptos fundamentales](#)

[Clase](#)

[Herencia](#)

[Objeto](#)

[Método](#)

[Evento](#)

[Atributos](#)

[Mensaje](#)

[Propiedad o atributo](#)

[Estado interno](#)

[Características de la POO](#)

[Abstracción](#)

[Encapsulamiento](#)

[Modularidad](#)

[Principio de ocultación](#)

[Polimorfismo](#)

[Herencia](#)

[Recolección de basura](#)

[CAPÍTULO IV](#)

[OTROS CONCEPTOS DE PROGRAMACIÓN](#)

[Socket](#)

[Scripting](#)

[CAPÍTULO V](#)

[ROBÓTICA](#)

[Introducción](#)

[Historia de la Robótica](#)

[Clasificación de robots](#)

[Según su cronología](#)

[Según su estructura](#)

[Robótica Educativa](#)

[Definición](#)

[Origen](#)

[Fases](#)

[Objetivos](#)

[CAPÍTULO VI](#)

[ARDUINO](#)

[Introducción a Arduino](#)

[Arduino Mega](#)

[Alimentación](#)

[Memoria](#)

[Comunicaciones](#)

[Programación](#)

[Reinicio Automático por Software](#)

[Módulos necesarios anexados a Arduino](#)

[Modulo Puente H L298](#)

[Comunicación vía Bluetooth](#)

[CAPÍTULO VII](#)

[DESARROLLO DE INGENIERÍA](#)

[DIAGRAMA DE BLOQUES](#)

[Descripción](#)

[CAPITULO VIII](#)

[ALICE](#)

[Introducción](#)

[Instalación de ALICE 2.4](#)

[Requerimientos](#)

[Descarga Alice](#)

[Instalación de Alice 2.4](#)

[Crear una aplicación en Alice](#)

[Escenarios](#)

[Objetos](#)

[Movimientos e interacción](#)

[Scripting](#)

[CAPITULO IX](#)

[SERVER SCRIPTING](#)

[CLIENTE SERVIDOR](#)

[Creando un servidor](#)

[Ejemplo comunicación bidireccional. Alice - Servidor](#)

[El servidor del proyecto Ana](#)

[CAPITULO X](#)

[CREACIÓN DEL ROBOT](#)

[CAPITULO XI](#)

[ARDUINO](#)

[Introduccion](#)

[Descarga Arduino](#)

[Conexiones Robot y Arduino](#)

[Programando Arduino](#)

[CAPITULO XII](#)

[EJEMPLO IMPLEMENTACIÓN EN CONJUNTO](#)

[RESULTADOS](#)

[CONCLUSIONES](#)

[BIBLIOGRAFÍA](#)

INTRODUCCIÓN

Problema de investigación

La demanda de Ingenieros en Argentina es mayor a la oferta, en los últimos años la cantidad de ingenieros recibidos aumento según Luis Caballero, Secretario de Políticas Universitarias, indicando que en el año 2014 egresaron 10.651 contra 6000 en el año 2012 pero a pesar de este alentador número, sigue en déficit el número de profesionales requeridos.

Muchas consultoras coinciden en que el número de nuevos ingenieros anuales rondan los 18.000, un número bastante alto, y como el término ingeniero involucra todas las carreras del país, basaremos nuestro estudio en el área de la informática, la rama de la ingeniería a la cual apunta este trabajo.

Siguiendo con la opinion de los que saben en el tema, obtuvimos un informe publicado por el diario Perfil en el año 2015, en el cual citaban una entrevista a Santiago Ceria el Director ejecutivo de la Fundación Sadosky donde se indicaba que la Argentina precisa unos 5000 profesionales del rubro informático por año y solamente estaban consiguiendo 3600 profesionales y muchos de ellos sin lograr recibirse, ya que son reclutados por las empresas antes de finalizar sus estudios y luego los abandonan.

Pero el déficit es todavía más grave: José María Louzao Andrade, presidente de la Cámara Argentina de Software y Servicios Informáticos, le comentó a este diario que “la demanda insatisfecha total de profesionales de tecnología, en el mercado, ronda los 15 mil expertos cada año”. Un numero bastante alarmante ya que se estaría buscando alternativas en el exterior para suplir esta falta de personal y en un país donde existen las Universidades en varias provincias, claramente hay algo que estamos haciendo mal al no lograr captar la atención de los egresados de los colegios secundarios.

También tenemos que contemplar que este rubro está en constante crecimiento, quizás el año que viene ya no sean 15.000 si no más los profesionales requeridos por año, debido a que a medida que avanza el tiempo, la tecnología avanza y a pasos agigantados.

Es por esto que se están abordando distintas políticas desde el estado y desde las casas de estudios avanzados como las Universidades o Terciarios para lograr conquistar este público.

Proposición

La propuesta de este trabajo final es lograr demostrar que un programa puede comunicarse con el mundo exterior, un robot. En este caso utilizaremos Alice, el cual es un software educativo e intentaremos realizar una comunicación en tiempo real a través de él y un robot, en este caso el robot lo construimos nosotros y elegimos que sea un auto de manera que sea sencillo y pueda ser realizado por un adolescente de nivel secundario.

La forma que el adolescente programara, es con su netbook del programa Conectar Igualdad o cualquiera que tenga acceso y el robot lo construiremos con una placa arduino, la cual se consigue en cualquier mercado de electrónica o a través de internet a un precio accesible.

El software antes mencionado se llama Alice, es gratuito, multiplataforma y de código libre. Fue creado para enseñar a programar por una Universidad de Estados Unidos y sus creadores pensaron en la manera más fácil para que un adolescente pueda iniciarse en el mundo de la programación sea divertida y no creando el típico “Hola Mundo”. En base a este software, realizaremos una comunicación en tiempo real con nuestro robot.

Este robot será creado con una placa Arduino y motores de corriente continua, motores económicos que pueden ser conseguidos en cualquier tienda electrónica o sacarlo a algún juguete viejo.

La comunicación en tiempo real será inalámbrica mediante Bluetooth por lo que creemos que le dará una sensación de control remoto y un plus a la motivación que buscamos encontrar.

Justificación del trabajo

Nuestra manera de creer que es posible un mundo con más ingenieros en informática es demostrar que no es algo de terminales y códigos extraños, nuestra tarea como educadores es intentar captar la atención de quienes no conocen mucho acerca del tema. Si logramos conseguir que nos escuchen podremos convencer a que se animen a estudiar estas carreras.

El software Alice ya fue creado para este propósito y podríamos sentarnos y ver si tiene o no efecto en su búsqueda de vocaciones IT, otro ejemplo de este tipo de programas es Scratch que también apunta al mismo sector.

La idea del proyecto Ana es darle una vuelta más y meterse de lleno en la robótica, demostrar que la robótica no es ciencia ficción y por el contrario, es una realidad que camina a pasos muy rápidos. Lograr conectar este software con algo del mundo exterior demuestra que un profesional del software no tiene barreras, más aún, con herramientas que están dentro de nuestro alcance.

Objetivos del proyecto

Nuestro propósito y único objetivo es lograr conseguir la atención de los adolescentes. Para lograrlo nosotros proponemos que al conseguir un valor agregado al software educativo Alice podremos estar por el camino indicado. El gobierno Nacional con la fundación Sadosky crearon el programa Dale Aceptar, el cual busca realizar que alumnos de colegios secundarios utilicen Alice, premiándolos con Play Station, Celulares, entre otras cosas.

Nosotros aprovechando que este software ya es conocido, decidimos aportar nuestro granito de arena, el cual se basa en comunicarlo con el mundo exterior, en este caso es un robot.

Para conseguir esta comunicación se va a tener que investigar acerca de las posibilidades que posee el software para intercambiar información.

Hemos considerado la posibilidad de ver su código fuente, ya que al ser Open Source podremos realizar una compilación nueva con nuestro soporte para una efectiva comunicación, es por esto, que tendremos un trabajo interesante en investigación para ver las distintas alternativas y analizar cuál es la más conveniente.

Luego tendremos que investigar acerca de Arduino, como poder crear un robot, en este caso hemos decidido crear un auto por la simplicidad en sus movimientos.

Esperamos tener un buen impacto al momento de mostrarle el trabajo al público que apunta y de esta manera poder asegurar que el objetivo está cumplido. Se podría solicitar en colegios o en Universidades alguna hora para lograr explicar la vocación. Así como se hace para las Expo educativas, donde explicamos acerca de las materias que posee la carrera, poder mostrar este trabajo entre otros, nos daría la pauta de si nuestro objetivo estuvo cerca o no, ver cuánto efecto causa en el público al cual apuntamos seria un buen parámetro para seguir profundizando en el tema.

MARCO TEÓRICO

CAPÍTULO I

ASIGNATURAS INVOLUCRADAS

Para lograr este proyecto se necesitó hacer un recorrido por varias materias cursadas durante la carrera Ingeniería en Informática en lo relacionado al desarrollo e implementación es decir, al software de este trabajo y también, investigar algunas correspondientes al área de la Ingeniería en Computación para abarcar los conceptos de robótica para lograr manipular la placa Arduino y de este modo conseguir construir un robot móvil.

Se tuvo que investigar sobre las placas Arduino, donde se tomo como base los conceptos enseñados en asignaturas Análisis de Circuitos y Circuitos Digitales, los cuales se tuvo que investigar a través de compañeros de la Universidad y docentes para un asesoramiento adecuado.

En el armado del Robot solo se tuvo que estudiar el funcionamiento del puente H L298 para la conexión de los motores, su aplicación con Arduino y la alimentación necesaria para que funcione correctamente. Luego el resto fue utilizar las salidas y entradas analógicas de Arduino para el ensamblado.

Luego para la parte de software bastó con los conceptos expuestos en asignaturas como Computación I y II, Programación II y los conceptos de Comunicación de Datos y Teleinformática para la parte de comunicación entre la computadora involucrada y el robot móvil.

CAPÍTULO II

ALICE

Introducción

Alice es [lenguaje de programación educativo libre y abierto](#), [orientado a objetos](#) con un [entorno de desarrollo integrado](#) (IDE). Está programado en [Java](#). Utiliza un entorno sencillo basado en «arrastrar y soltar» para crear [animaciones](#) mediante [modelos 3D](#). Este [software](#) fue desarrollado por los investigadores de la [Universidad Carnegie Mellon](#), entre los que destaca [Randy Pausch](#).

Las versiones de Alice pueden ejecutarse en [Microsoft Windows](#), [Mac OS X](#) y [Linux](#).

Alice es un entorno de programación 3D innovador que hace que sea fácil crear una animación para contar una historia, jugando un juego interactivo, o un video para compartir en la web. Alice es una herramienta de enseñanza libre disposición diseñada para ser la primera exposición de un estudiante de programación orientada a objetos. Permite a los estudiantes a aprender los conceptos fundamentales de programación en el contexto de la creación de películas de animación y video juegos simples. En Alice, objetos 3-D (por ejemplo, personas, animales y vehículos) poblar un mundo virtual y los estudiantes crean un programa para animar los objetos.

En la interfaz interactiva de Alice, los estudiantes que arrastrar y soltar los azulejos gráficos para crear un programa, según las instrucciones que se corresponden con las declaraciones habituales en un lenguaje de programación orientado a la producción, tales como Java, C++ y C#. Alice permite a los estudiantes para ver de inmediato cómo se ejecutan sus programas de animación, lo que les permite entender fácilmente la relación entre las instrucciones de programación y el comportamiento de los objetos en su animación. Mediante la manipulación de los objetos en su mundo virtual, los estudiantes adquieren experiencia con todas las construcciones de programación normalmente se enseñan en un curso de introducción a la programación.

Objetivo de Alice

El programa se desarrolló prioritariamente para solucionar tres problemas fundamentales del software educativo:

La mayoría de los lenguajes de programación están diseñados para producir otros programas, cada vez más complejos. Alice está diseñado únicamente para enseñar a programar.

Alice está íntimamente unido a su IDE. No hay que recordar ninguna sintaxis especial. De todas formas, acepta tanto el modelo de [programación orientada a objetos](#) como la [dirigida a eventos](#).

Alice está diseñado para el público que normalmente no se enfrenta a problemas de programación, tales como alumnos de secundaria, mediante un sistema de “arrastrar y soltar”.

Al no existir en el entorno de programación Alice un editor del código de programación propiamente dicho, se sortean las dificultades inherentes al rigor sintáctico a los primeros pasos en la programación orientada a objetos. Se trata de que el árbol no nos impida ver el bosque. Algunos profesores han encontrado que estudiantes que pueden programar en Alice se enfrentan a posteriori con dificultades a la hora de introducirse en un lenguaje de programación tradicional que usa editor de texto (para la sintaxis). En un posible itinerario de aprendizaje, así como Scratch es el paso previo natural a Alice por su sencillez de uso, [Greenfoot](#) puede ser de utilidad a la hora de afrontar la transición a un lenguaje de programación orientada a objetos en un entorno de desarrollo con editor de texto. Greenfoot mantiene el atractivo visual y amigabilidad del entorno, así como la orientación a la realización de -entre otras cosas- videojuegos. Este inconveniente también puede ser sorteado gracias a la cantidad de editores que auto-completan la sintaxis, proveyendo las funciones para el módulo importado, lenguaje utilizado o palabras claves del mismo.

En estudios realizados en el IthacaCollege y en [Saint Joseph'sUniversity](#), las notas medias de estudiantes sin experiencia de programación y en su primer curso de informática subieron de suficiente a notable y su retención del 47% al 88%.

TICS

Las tecnologías de la información (TICs) últimamente llamadas como nuevas tecnologías de la información (NTICS) han experimentado una gran expansión estos últimos años. Es un concepto que está estrechamente vinculado con la informática, aporta una serie de ventajas y funcionalidades que vale la pena mencionar, al menos algunas de las más importantes, como lo son: la gran capacidad de tratamiento y almacenamiento de la información, interactividad y automatización de tareas, acceso flexible a la información y fácil transporte de datos, canales de comunicación, integración de medios y códigos, reducción de costes, tiempo y esfuerzo en la realización de trabajos, etc.

Esto se debe a la digitalización de la información, los nuevos actores electrónicos como Tablet, celulares inteligentes, robots, lectores de libros electrónicos, consolas de video-juegos, fotografía digital, entre otros, desplacen a los antiguos protagonistas como manuales, apuntes, telégrafos, procesadores de textos, etc.

Estos nuevos soportes, en un futuro inmediato, serán utilizados en todos los niveles de educativos o al menos se espera.

En cualquier caso, el desarrollo de estas nuevas tecnologías no tiene por qué suplantar el papel del docente en cuanto a la responsabilidad en el diseño y la “orientación” del aprendizaje y la instrucción, sino que, es más, pueden representar una herramienta valiosísima en la medida que ayuden al nuevo alumno a desarrollar su conocimiento desde las nuevas premisas del aprendizaje.

Hoy en día hablar de TIC's, como las nuevas tecnologías antes mencionadas se distancia mucho de lo que era el manejo operativo de una computador o un procesador de textos, debemos apuntar a este nuevo paradigma que está en constante evolución y crece a escalas no lineales. Se debe dejar de inculcar en los chicos el hecho de que la informática es la ofimática y sí hacer hincapié en la programación como base elemental. Adrián Paenza, doctor en ciencias matemática por la Facultad de Ciencias Exactas y Naturales en la Universidad de Buenos Aires, en reiteradas ocasiones el sugería que había que enseñar a programar en los colegios, el y otros destacados en la educación hicieron que esto en el año 2015 sea escuchado por el Consejo Federal de Educación y como propuesta hacia la nueva alfabetización enseñaran a programar en los colegios públicos. La implementación de esta medida al momento de redactar el presente informe no estaba definida, lo que sí se puede mencionar es que esta medida debe acompañar al Plan Nacional

de Inclusión Digital que está vinculado con Program.ar. Con esta medida se busca llevar a todos los planes de estudios de colegios primarios y secundarios mecanismos diversos de enseñanza en lo que respecta al desarrollo del software. Busca promover la creatividad de los alumnos y motivarlos a que sigan carreras relacionadas para conseguir la demanda de profesionales en el rubro y romper con los mitos acerca de lo que es y lo que no es la informática.

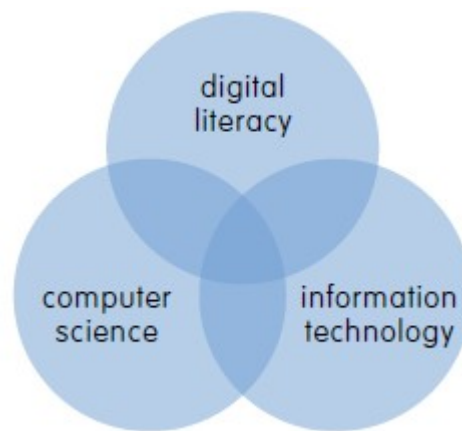
En concordancia con lo que estamos abordando, si bien las TICS involucran más que la programación, enseñar a programar sería el “abc” de la informática hoy, ya no alcanza con saber usarlas.

En el año 2015 El Consejo Federal de Educación declaró la programación de software como una herramienta de enseñanza estratégica

Una consecuencia es que la informática es a menudo olvidado o ignorado en el encabezamiento de las ‘TIC’, lo que resulta que en la enseñanza está sesgada hacia "cómo utilizar el software de oficina "en lugar de los conocimientos que constituirá la base para el resto de la vida del alumno. Esto ha llevado a que muchas personas tengan una visión muy negativa de ' TIC '.

Fundamentalmente, la industria, las escuelas y los profesores carecen de un lenguaje coherente que regula cómo comunicar sus necesidades de negocio, asesoramiento profesional y el contenido curricular de los interesados de la materia incluyendo al Gobierno. La terminología existente da lugar a una gran confusión que puede afectar elección de asignaturas de los jóvenes y dar lugar a una mala formulación de políticas.

Debido a la connotación negativa del término TIC se propone desglosar y hacer más abarcativa la definición al proponer:



Esta división sugiere tres conceptos a saber:

1- Digital Literacy, su traducción al español es alfabetización digital es la habilidad que posee el profesional para localizar, organizar, entender, evaluar y analizar información utilizando las tecnologías modernas o también llamadas tecnologías digitales. Implica tanto el conocimiento de cómo trabaja la alta tecnología de hoy en día como así también la comprensión de cómo puede ser utilizada. Las personas digitalmente alfabetizados pueden comunicarse y trabajar más eficientemente especialmente con aquellos que poseen los mismos conocimientos y habilidades.

Los desafíos que propone la educación hoy en colegios y universidades son muchos, en varios países están trabajando duro para ofrecer alta calidad en la educación y para convertirse en instituciones atractivas, innovadoras y socialmente útiles. El uso de herramientas digitales se considera importante para este logro. Por ejemplo, el objetivo de un estudio a este respecto es examinar cómo la gestión universitaria puede estimular la enseñanza de académicos para continuar e incrementar el uso de las TIC en la enseñanza y el aprendizaje. Debido a la continua introducción de nuevas tecnologías, los cambios en la educación superior están en curso, por lo tanto, la identificación de formas de motivar a los profesores para utilizar las TIC se ha convertido en una preocupación importante para la gestión universitaria.

2- Information technology o su traducción al español como Tecnología de la Información (T) es la aplicación de computadoras y equipos de telecomunicación para almacenar, recuperar, transmitir y manipular datos, con frecuencia utilizado en el contexto de las empresas y meramente educativos. El término es comúnmente utilizado como

sinónimo de computadoras o redes de comunicación pero también abarca otras tecnologías de distribución de información, tales como tablets, televisores inteligentes, teléfonos inteligentes, etc. Múltiples industrias están asociadas con las tecnologías de la información, incluyendo hardware y software de las computadoras, electrónica, semiconductores, internet, equipos de telecomunicación, comercio electrónico y servicios computacionales.

3- Computer Science o ciencias de la computación está relacionada con los conceptos teóricos, investigaciones acerca de nuevos paradigmas a nivel software o hardware, donde los métodos se van renovando. Entre ellos podemos encontrar la Robótica como una rama que emerge a pasos agigantados tanto a nivel empresarial como educativo y es por esto que debemos abordar y crear las herramientas para que la educación pueda estar a la altura de los tiempos.

A continuación vemos una serie de puntos que se deben mejorar para poder encarar este nuevo desafío.

Hay una escasez de profesores especializados capaces para enseñar Informática

Hay una falta de desarrollo profesional continuo (DPC) para los maestros.

La infraestructura escolar está frenando la buena enseñanza.

Recursos didácticos técnicos son a menudo inadecuados.

Informática es una disciplina académica rigurosa y debe ser reconocido como tal en las escuelas.

Cada niño debe tener la oportunidad de aprender computación en la escuela.

Existe la necesidad de cualificaciones en los aspectos de la computación que se puede acceder en la escuela pero no se enseña actualmente. También hay una necesidad de métodos de evaluación inadecuados, existentes a ser actualizados.

Es necesario realizar actividades de mejora y enriquecimiento (E & E) para apoyar el plan de estudios.

CAPÍTULO III

PROGRAMACIÓN ORIENTADA A OBJETOS

Programación en clases y objetos

La programación orientada a objetos o POO (OOP según sus siglas en inglés) es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo, herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos.

Los objetos son entidades que tienen un determinado estado, comportamiento (método) e identidad:

El estado está compuesto de datos o informaciones; serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).

El comportamiento está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.

La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a alguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una programación estructurada camuflada en un lenguaje de programación orientado a objetos.

La POO difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean Programación Orientada a Objetos, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

Origen

Los conceptos de la programación orientada a objetos tienen origen en Simula 67, un lenguaje diseñado para hacer simulaciones, creado por Ole-Johan Dahl y Kristen Nygaard, del Centro de Cómputo Noruego en Oslo. En este centro se trabajaba en simulaciones de naves, que fueron confundidas por la explosión combinatoria de cómo las diversas cualidades de diferentes naves podían afectar unas a las otras. La idea surgió al agrupar los diversos tipos de naves en diversas clases de objetos, siendo responsable cada clase de objetos de definir sus propios datos y comportamientos. Fueron refinados más tarde en Smalltalk, desarrollado en Simula en Xerox PARC (cuya primera versión fue escrita sobre Basic) pero diseñado para ser un sistema completamente dinámico en el cual los objetos se podrían crear y modificar "sobre la marcha" (en tiempo de ejecución) en lugar de tener un sistema basado en programas estáticos.

La programación orientada a objetos se fue convirtiendo en el estilo de programación dominante a mediados de los años ochenta, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las Interfaces gráficas de usuario, para las cuales la programación orientada a objetos está particularmente bien adaptada. En este caso, se habla también de programación dirigida por eventos.

Las características de orientación a objetos fueron agregadas a muchos lenguajes existentes durante ese tiempo, incluyendo Ada, BASIC, Lisp y Pascal, entre otros. La adición de estas características a los lenguajes que no fueron diseñados inicialmente para ellas condujo a menudo a problemas de compatibilidad y en la capacidad de mantenimiento del código. Los lenguajes orientados a objetos "puros", por su parte, carecían de las características de las cuales muchos programadores habían venido a depender. Para saltar este obstáculo, se hicieron muchas tentativas para crear nuevos lenguajes basados en métodos orientados a objetos, pero permitiendo algunas características imperativas de maneras "seguras". El Eiffel de Bertrand Meyer fue un temprano y moderadamente acertado lenguaje con esos objetivos, pero ahora ha sido esencialmente reemplazado por Java, en gran parte debido a la aparición de Internet y a la implementación de la máquina virtual de Java en la mayoría de navegadores. PHP en su versión 5 se ha modificado; soporta una orientación completa a objetos, cumpliendo todas las características propias de la orientación a objetos.

Conceptos fundamentales

La programación orientada a objetos es una forma de programar que trata de encontrar una solución a estos problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

Clase

Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

Herencia

(Por ejemplo, herencia de la clase C a la clase D) es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables públicas declaradas en C. Los componentes registrados como "privados" (private) también se heredan, pero como no pertenecen a la clase, se mantienen escondidos al programador y sólo pueden ser accedidos a través de otros métodos públicos. Esto es así para mantener hegemónico el ideal de POO.

Objeto

Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa). Es una instancia a una clase.

Método

Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

Evento

Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.

Atributos

Características que tiene la clase

Mensaje

Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

Propiedad o atributo

Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

Estado interno

Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

Componentes de un objeto

Atributos, identidad, relaciones y métodos.

Identificación de un objeto

Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

En comparación con un lenguaje imperativo, una "variable" no es más que un contenedor interno del atributo del objeto o de un estado interno, así como la "función" es un procedimiento interno del método del objeto.

Características de la POO

Existe un acuerdo acerca de qué características contempla la "orientación a objetos". Las características siguientes son las más importantes:

Abstracción

Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos, y, cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

Encapsulamiento

Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

Modularidad

Se denomina modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas.

Principio de ocultación

Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no puedan cambiar el estado interno de un objeto de manera inesperada, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

Polimorfismo

Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

Herencia

Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple.

Recolección de basura

La recolección de basura o garbage collector es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando.

CAPÍTULO IV

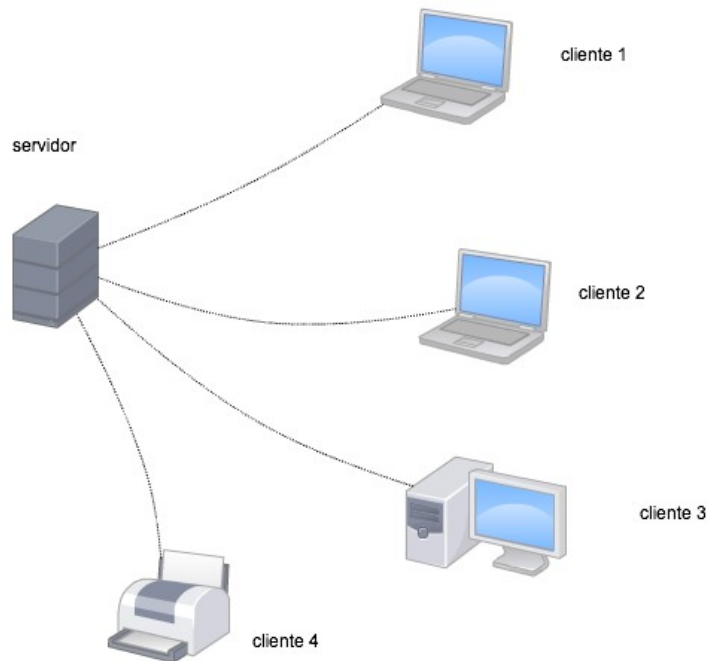
OTROS CONCEPTOS DE PROGRAMACIÓN

Socket

Antes de abordar en que es un socket, debemos dar una introducción a lo que hoy ya todos conocemos pero quizás no asociamos que es el cliente-servidor.

El paradigma Cliente/Servidor es quizás el más conocido de los paradigmas para aplicaciones de red. Se usa para describir un modelo de interacción entre dos procesos, que se ejecutan de forma simultánea. Este modelo es una comunicación basada en una serie de preguntas y respuestas, que asegura que si dos aplicaciones intentan comunicarse, una comienza la ejecución y espera indefinidamente que la otra le responda y luego continua con el proceso.

Tenemos que identificar que el par cliente-servidor contiene a dos participantes pero que la realidad es que los servidores esperan muchas conexiones de clientes por lo que un esquema de comunicación tipo podría ser el siguiente:



Una de las técnicas de comunicación entre procesos a través de alguna red puede ser mediante Socket, es un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada, dependiendo del protocolo de comunicación, entre los que se destacan los más comunes como UDP o TCP, según se necesite orientados o no a conexión.

El término socket es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.

Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

El funcionamiento para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro, es decir, saber en qué IP y puerto puede ser encontrado.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de octetos, es decir, datos relevantes a su finalidad, hablar en un lenguaje que sepan que significa. Esto quiere decir que si un proceso envía una secuencia de bits tales como 11110000, la otra parte sepa decodificar y realizar algún procedimiento en base a dicho mensaje.

Siguiendo con lo que mencionamos, son necesarios los dos recursos que originan el concepto de socket:

Un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota.

Un par de números de puerto, que identifican a un programa dentro de cada computadora.

Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los programas que se denomina programa "cliente". El segundo programa espera a que otro inicie la comunicación, por este motivo se denomina programa "servidor".

Un socket es un proceso o hilo existente en la máquina cliente y en la máquina servidora, que sirve en última instancia para que el programa servidor y el cliente lean y escriban la información. Esta información será la transmitida por las diferentes capas de red.

Existe una variante de los sockets denominada Unix domain sockets, o bien, "interprocess communication sockets" (IPC sockets). Éstos se encuentran especificados en la norma POSIX y tienen como propósito la intercomunicación entre programas dentro de la misma computadora, facilitando así la optimización de recursos para este caso en concreto.

Scripting

Un script, archivo de órdenes, archivo de procesamiento por lotes o guión, es un programa generalmente bastante sencillo, que es almacenado en un archivo de texto plano. Son casi siempre interpretados pero a diferencia de los programas interpretados, estos últimos no siempre son

scripts. El uso de estos script es la de llevar a cabo diversas tareas como combinar componentes, interactuar con el sistema operativo, entre distintos script o con el usuario.

CAPÍTULO V

ROBÓTICA

Introducción

La robótica es la rama de la tecnología que se dedica al diseño, construcción, operación, disposición estructural, manufactura y aplicación de los robots.^{1 2} La robótica combina diversas disciplinas como son: la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física.³ Otras áreas importantes en robótica son el álgebra, los autómatas programables, la animatrónica y las máquinas de estados.

El término robot se popularizó con el éxito de la obra RUR (Robots Universales Rossum), escrita por Karel Čapek en 1920. En la traducción al inglés de dicha obra, la palabra checa robota, que significa trabajos forzados, fue traducida al inglés como robot.⁴

Historia de la Robótica

La historia de la robótica va unida a la construcción de "artefactos", que trataban de materializar el deseo humano de crear seres a su semejanza y que lo descargasen del trabajo. El ingeniero español Leonardo Torres Quevedo (GAP) (que construyó el primer mando a distancia para su automóvil mediante telegrafía sin hilo, el ajedrecista automático, el primer transbordador aéreo y otros muchos ingenios) acuñó el término "automática" en relación con la teoría de la automatización de tareas tradicionalmente asociadas.

Karel Čapek, un escritor checo, acuñó en 1921 el término "Robot" en su obra dramática Rossum's Universal Robots / R.U.R., a partir de la palabra checa robota, que significa servidumbre o trabajo forzado. El término robótica es acuñado por Isaac Asimov, definiendo a la ciencia que estudia a los robots. Asimov creó también las Tres Leyes de la Robótica. En la ciencia ficción el hombre ha imaginado a los robots visitando nuevos

mundos, haciéndose con el poder, o simplemente aliviando de las labores caseras.

Clasificación de robots

Según su cronología

La que a continuación se presenta es la clasificación más común:

1ª Generación.

Manipuladores. Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.

2ª Generación.

Robots de aprendizaje. Repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano. El modo de hacerlo es a través de un dispositivo mecánico. El operador realiza los movimientos requeridos mientras el robot le sigue y los memoriza.

3ª Generación.

Robots con control sensorizado. El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

4ª Generación.

Robots inteligentes. Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso. Esto permite una toma inteligente de decisiones y el control del proceso en tiempo real.

Según su estructura

La estructura, es definida por el tipo de configuración general del Robot, puede ser metamórfica. El concepto de metamorfismo, de reciente aparición, se ha introducido para incrementar la flexibilidad funcional de un Robot a través del cambio de su configuración por el propio Robot. El metamorfismo admite diversos niveles, desde los más elementales (cambio de herramienta o de efecto terminal), hasta los más complejos como el cambio o alteración de algunos de sus elementos o subsistemas estructurales. Los dispositivos y mecanismos que pueden agruparse bajo la denominación genérica del Robot, tal como se ha indicado, son muy

diversos y es por tanto difícil establecer una clasificación coherente de los mismos que resista un análisis crítico y riguroso. La subdivisión de los Robots, con base en su arquitectura, se hace en los siguientes grupos: poliarticulados, móviles, androides, zoomórficos e híbridos.

1. Poliarticulados

En este grupo se encuentran los Robots de muy diversa forma y configuración, cuya característica común es la de ser básicamente sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas, y con un número limitado de grados de libertad. En este grupo, se encuentran los manipuladores, los Robots industriales, los Robots cartesianos y se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo.

2. Móviles

Son Robots con gran capacidad de desplazamiento, basada en carros o plataformas y dotada de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores. Estos Robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

3. Androides

Son Robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Actualmente, los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, fundamentalmente, al estudio y experimentación. Uno de los aspectos más complejos de estos Robots, y sobre el que se centra la mayoría de los trabajos, es el de la locomoción bípeda. En este caso, el principal problema es controlar dinámicamente y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del Robot.

4. Zoomórficos

Los Robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos. A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los Robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los Robots zoomórficos no caminadores está muy poco evolucionado. Los experimentos efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. Los Robots zoomórficos caminadores múltipedos son muy numerosos y están siendo objeto de experimentos en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, piloteados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos Robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

5. Híbridos

Corresponden a aquellos de difícil clasificación, cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo, uno de los atributos de los Robots móviles y de los Robots zoomórficos.

Robótica Educativa

Dado el carácter polivalente y multidisciplinario de la Robótica Educativa, ésta ayuda en el desarrollo e implantación de una nueva cultura tecnológica en todas las regiones del país, permitiendo el entendimiento, mejoramiento y desarrollo de sus propias tecnologías, y es una experiencia que contribuye al desarrollo de la creatividad y el pensamiento de los estudiantes.

Uno de los principales objetivos de la Robótica Educativa, es la generación de entornos de aprendizaje basados fundamentalmente en la actividad de los estudiantes. Es decir, ellos podrán concebir, desarrollar y poner en práctica diferentes Robots educativos que les permitirán resolver algunos problemas y les facilitarán al mismo tiempo, ciertos aprendizajes.

En otras palabras, se trata de crear las condiciones de apropiación de conocimientos y permitir su transferencia en diferentes campos del conocimiento.

Se puede concluir que la Robótica Educativa se ha desarrollado como una perspectiva de acercamiento a la solución de problemas derivados de distintas áreas del conocimiento como las matemáticas, las ciencias naturales y experimentales, la tecnología y las ciencias de la información y la comunicación, entre otras. Uno de los factores más interesantes es que la integración de diferentes áreas se da de manera natural.

En efecto, la construcción de un Robot educativo requiere del conocimiento de diversas áreas. Por mencionar algunas, es necesario tener conocimientos de mecánica para poder construir la estructura del Robot. También se requieren conocimientos de electricidad para poder animar desde el punto de vista eléctrico al Robot. Asimismo, es importante tener conocimientos de electrónica para poder dar cuenta de la comunicación entre el computador y el Robot. Finalmente, es necesario tener conocimientos de informática para poder desarrollar un programa en cualquier lenguaje de programación que permita controlar al Robot.

Es aquí justamente, que la Robótica Educativa muestra una de sus principales bondades, al permitir integrar distintas áreas del conocimiento, en un proyecto que requiere de un buen ejercicio de integración y que en este caso, la construcción misma de un Robot educativo, es un excelente pretexto para lograr esta integración desde el punto de vista cognitivo y tecnológico. En otras palabras, el conocimiento no puede estar atomizado o fraccionado. Es necesario integrarlo en el momento del desarrollo del Robot educativo.

Mediante la integración de diferentes áreas de conocimiento, los estudiantes adquieren habilidades generales y nociones científicas, involucrándose en un proceso de resolución de problemas con el fin de desarrollar en ellos, un pensamiento sistémico, estructurado, lógico y formal.

Uno de los grandes retos de la Robótica Educativa, es demostrar que los estudiantes pueden construir sus propias representaciones y conceptos de la ciencia y de la tecnología, mediante la utilización, manipulación y control de ambientes de aprendizajes robotizados, a través de la solución

de problemas concretos. De esta manera, los proyectos se tornan significativos para ellos.

El desarrollo de situaciones de aprendizaje en Robótica Educativa necesita que los objetivos de aprendizaje no sean enunciados a priori, que el material sea dado para ser manipulado y observado. Se hace hincapié sobre el proceso de construcción y adquisición de conceptos. Es a través de la manipulación y la exploración que el estudiante va a dirigir y a centrar sus percepciones y observaciones. Cuando esta manipulación es efectuada por el profesor, éste debe según Gagné (1976) dirigir y centrar la atención del estudiante. Aquí, es el desarrollo de la experiencia quien impone la dirección de las observaciones.

Definición

Es el conjunto de actividades pedagógicas que apoyan y fortalecen áreas específicas del conocimiento y desarrollan competencias en el alumno, a través de la concepción, creación, ensamble y puesta en funcionamiento de robots.

El objetivo de la enseñanza de la Robótica, es lograr una adaptación de los alumnos a los procesos productivos actuales, en donde la Automatización (Tecnología que está relacionada con el empleo de sistemas mecánicos, electrónicos y basados en computadoras; en la operación y control de la producción) juega un rol muy importante. Sin embargo la robótica se considera un sistema que va más allá de una aplicación laboral.

Algo que también cabe mencionar en el estudio de la Robótica, es la gran necesidad de una perfecta relación entre el Software y el Hardware del Robot, ya que los movimientos que realizará éste Robot es un acoplamiento entre lo físico y lo lógico.

Origen

La Robótica Educativa se centra principalmente en la creación de un robot con el único fin de desarrollar de manera mucho más práctica y didáctica las habilidades motoras y cognitivas de quienes los usan. De esta manera se pretende estimular el interés por las ciencias duras y motivar la actividad sana. Así mismo hacer que el niño logre una organización en grupo, discusiones que permitan desarrollar habilidades sociales, respetar cada uno su turno para exponer y aprender a trabajar en equipo.

Fases

Se tiene la idea de que se construye un robot utilizando cables y equipo para hacerlo en la vida real, pero no es así, porque en la Robótica Educativa se pretende inicialmente crear un robot en computador, se hace en programas especiales como el xLogo (usando en verdad, una versión libre de éste), donde se realiza un pequeño estudio que ve si éste robot es realizable o no en la realidad. Aquí, al tenerlo en el computador se establece la función que cumplirá este robot, las cuales son específicas para realizar pequeñas tareas (como traer objetos o limpiar cosas, por ejemplo), y se observa en la pantalla el cómo se ve este robot. Luego, eliminando y arreglando, se procede a utilizar materiales para llevarlo a cabo en la realidad.

En este punto, se utilizan variados materiales, pueden ser desde piezas de sistemas constructivos como Lego, Múltiplo o Robo-Ed, a materiales de desecho que no se ocupan en casa (como cajas de cartón y circuitos en desuso). Aunque, también se usan materiales más de clase como son metales u otros derivados.

Objetivos

Que sean más ordenados.

Promover los experimentos, donde el equivocarse es parte del aprendizaje y el autodescubrimiento.

Ser más responsables con sus cosas.

Desarrollar mayor movilidad en sus manos.

Desarrollar sus conocimientos.

Desarrollar la habilidad en grupo, permitiendo a las personas socializar.

Desarrollar sus capacidades creativas.

Poder observar cada detalle.

Desarrollar el aprendizaje en forma divertida.

Glosario de términos utilizados en robótica

Actuador: Dispositivo que produce algún tipo de movimiento a partir de una orden proveniente de la interfaz.

Electroimán: Dispositivo que se magnetiza cuando se hace circular por el una corriente eléctrica. Se utiliza mucho para producir movimientos por medio de señales eléctricas.

Entrada de Sensor: Terminal de la interfaz en la que se pueden conectar sensores de distintos tipos.

Interfaz: Puente entre el sistema a controlar y el ordenador. Su función es transformar señales bajas en señales de mayor capacidad.

LED: Diodo emisor de luz.

Lenguaje computadora: Programa mediante el cual se puede especificar una serie de instrucciones para que la computadora pueda realizar una serie de tareas de forma independiente.

Programa de Control: Conjunto de instrucciones que están situadas en la computadora y determinan la función del mecanismo que se controla(robot).

Puerto: Enchufe de la computadora en donde se pueden conectar diferentes tipos de dispositivos.

Robot: Término derivado del vocablo checo Robota (trabajo, prestación personal). Máquina que gracias a un tipo de programación puede realizar tareas específicas.

Sensor: Dispositivo que proporciona información a la computadora de lo que ocurre en el entorno o en el robot que está siendo controlado.

Materiales utilizados en robótica educativas

En entornos de robótica educativa y de ocio se utilizan con frecuencia unos dispositivos denominados interfaces de control, o más coloquialmente controladoras,² cuya misión es reunir en un solo elemento todos los sistemas de conversión y acondicionamiento que necesita un ordenador personal PC para actuar como cerebro de un sistema de control automático o de un robot. Las interfaces de control se podrían así definir como placas multifunción de E/S (entrada/salida) en configuración externa (es decir, no son placas instalables en ninguna bahía de expansión del PC), que se conectan con el PC mediante alguno de los puertos de comunicaciones propios del mismo (paralelo, serie o USB,

generalmente) y sirven de interfaz entre el mismo y los sensores y actuadores de un sistema de control. Las interfaces proporcionan, de forma general, una o varias de las siguientes funciones:

Entradas analógicas, que convierten niveles analógicos de voltaje o de corriente en información digital procesable por el ordenador. A este tipo de entradas se pueden conectar distintos sensores analógicos, como por ejemplo una LDR (resistencia dependiente de la luz).

Salidas analógicas, que convierten la información digital en corriente o voltaje analógicos de forma que el ordenador pueda controlar sucesos del "mundo real". Su principal misión es la de excitar distintos actuadores del equipamiento de control: válvulas, motores, servomecanismos, etc.

Entradas y salidas digitales, usadas en aplicaciones donde el sistema de control sólo necesita discriminar el estado de una magnitud digital (por ejemplo, un sensor de contacto) y decidir la actuación o no de un elemento en un determinado proceso, por ejemplo, la activación/desactivación de una electroválvula.

recuento y temporización, algunas tarjetas incluyen este tipo de circuitos que resultan útiles en el recuento de sucesos, la medida de frecuencia y amplitud de pulsos, la generación de señales y pulsos de onda cuadrada, y para la captación de señales en el momento preciso.

Algunas de las interfaces de control más avanzadas cuentan además con la electrónica precisa para el acondicionamiento y la conversión de las señales, con sus propios microprocesador y memoria. Así, son capaces hasta de almacenar pequeños programas de control transmitidos desde un PC que luego pueden ejecutar independientemente de su conexión a éste.

Algunas de ellas disponen también de bibliotecas de programación de las E/S para permitir su utilización con distintos lenguajes de propósito general, entre ellos: LOGO, BASIC y C. Existen varios modelos comerciales, entre los que se pueden mencionar:

Interfaz FlowGo, de Data Harvest

Interfaz ROBO TX Controller de fischertechnik

Ladrillo RCX, de Lego

Interfaz Enconor, de Enconor Tecnología Educativa

Robot Programable Moway, de Minirobots

Sistema constructivo Multiplo, de RobotGroup

Kits educativos y contenidos Robo-Ed [1]

CAPÍTULO VI

ARDUINO

Introducción a Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (bootloader) que corre en la placa.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software del ordenador (por ejemplo: Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles muy fáciles de usar, debido a que el IDE con el que trabaja es fácil de aprender a utilizar, y el lenguaje de programación con el que trabaja es simple, pues se creó para artistas, diseñadores, aficionados y cualquier interesado en crear entornos u objetos interactivos. Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP). Las placas pueden ser hechas a mano o compradas montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño

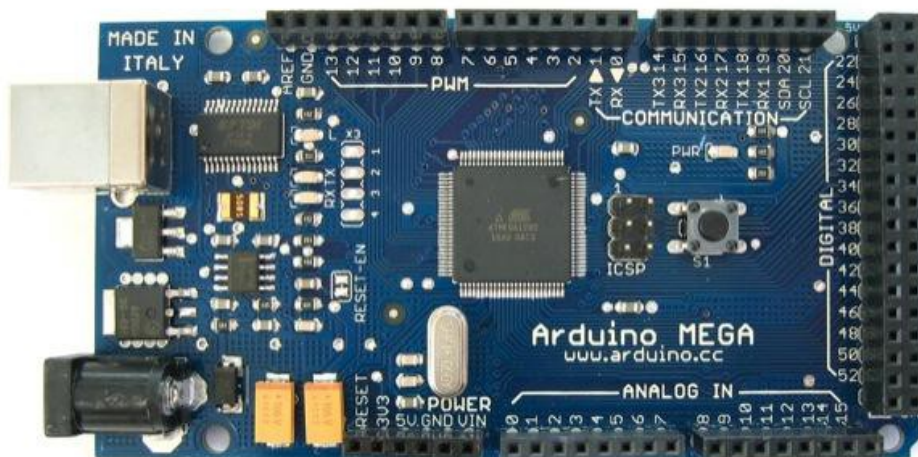
de referencia (CAD) están disponibles bajo una licencia abierta, así pues eres libre de adaptarlos a tus necesidades.

El proyecto Arduino recibió una mención honorífica en la categoría de Comunidades Digital en el PrixArs Electrónica de 2006.

Arduino Mega

El Arduino Mega es una placa microcontrolador basado ATmega1280. Tiene 54 entradas/salidas digitales (de las cuales 14 proporcionan salida PWM), 16 entradas digitales, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reset. Contiene todo lo necesario para hacer funcionar el microcontrolador; simplemente conéctalo al ordenador con el cable USB o aliméntalo con un transformador o batería para empezar.

Imagen de una placa Arduino Mega 2560



Características de Arduino Mega 2560

Microcontrolador ATmega1280

Voltaje de funcionamiento 5V

Voltaje de entrada (recomendado) 7-12V

Voltaje de entrada (limite) 6-20V

Pines E/S digitales 54 (14 proporcionan salida PWM)

Pines de entrada analógica 16

Intensidad por pin 40 mA

Intensidad en pin 3.3V 50 mA

Memoria Flash 256 KB de las cuales 4 KB las usa el gestor de arranque (bootloader)

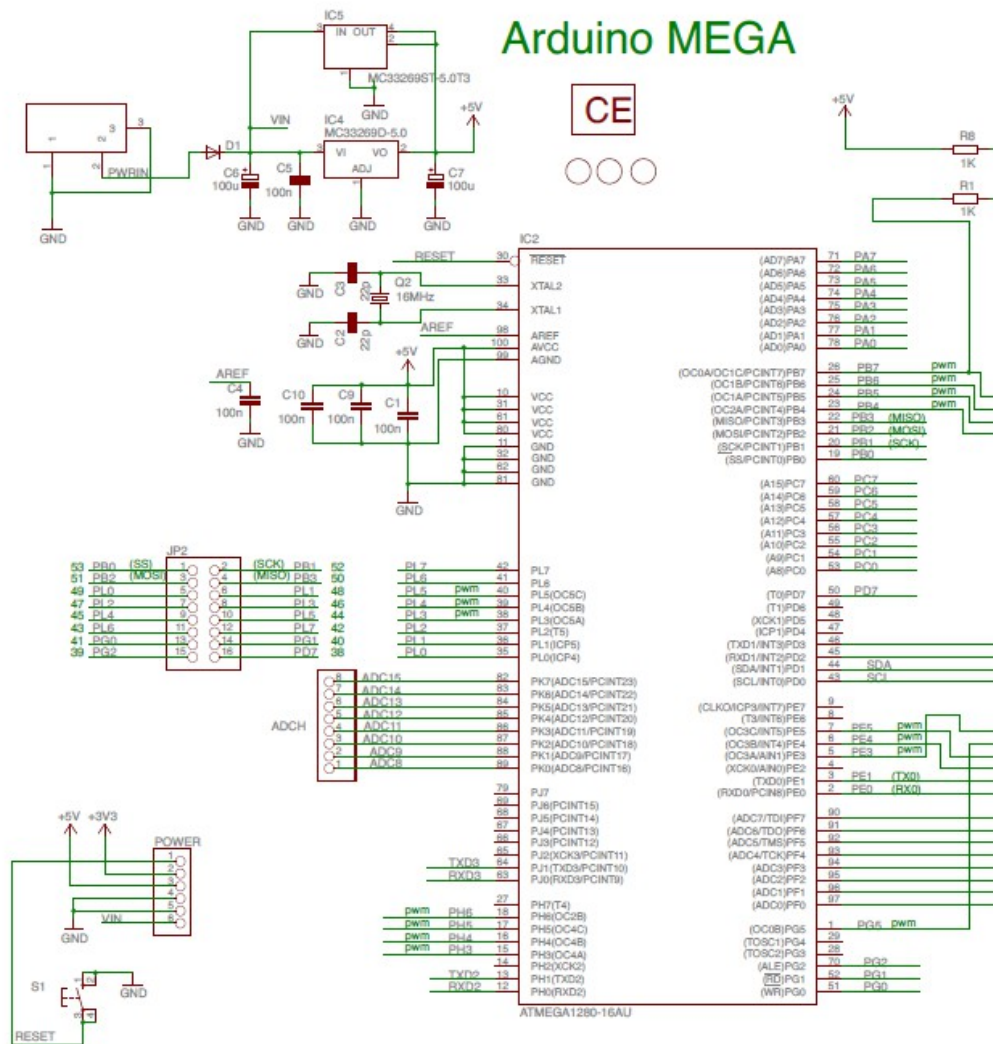
SRAM 8 KB

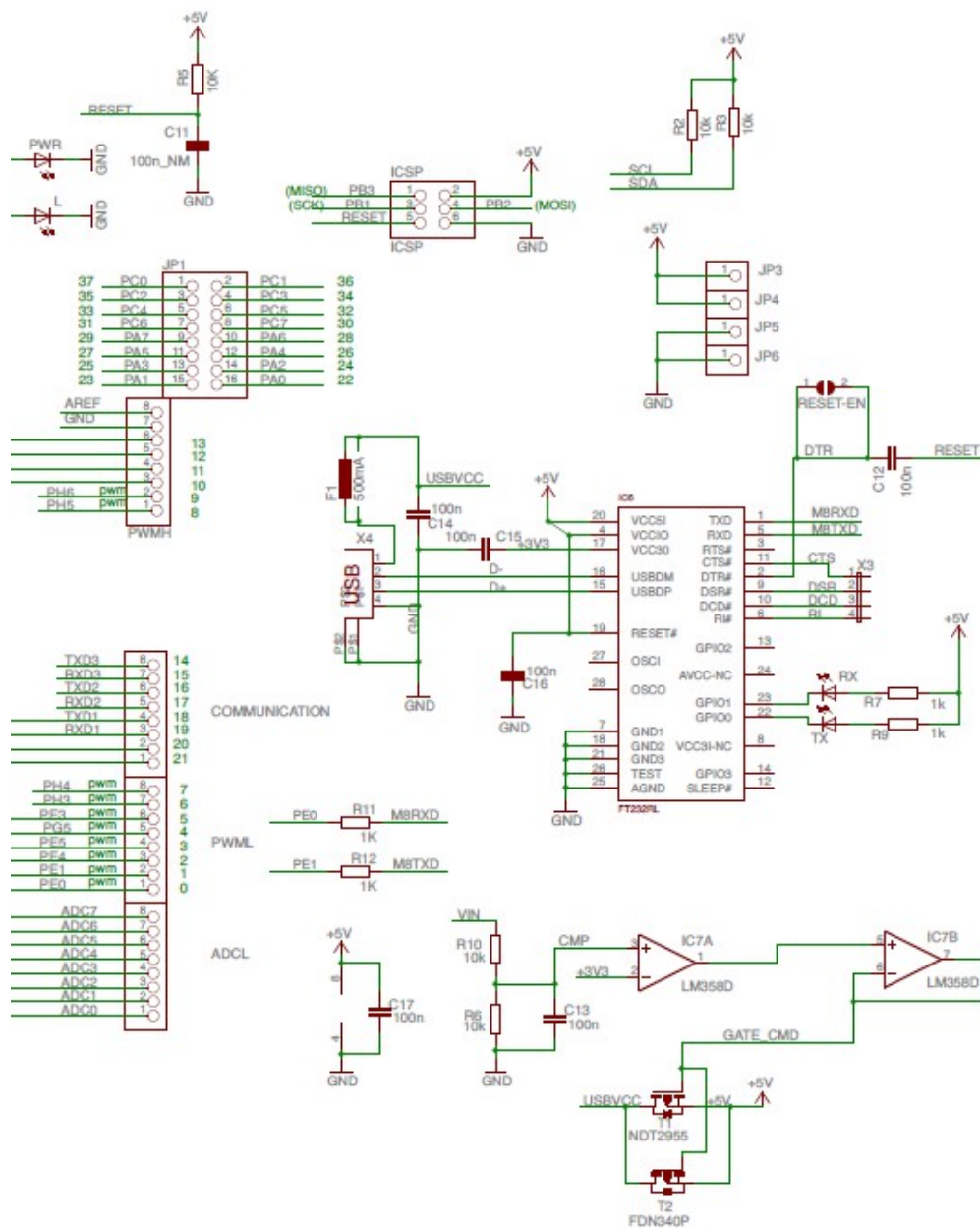
EEPROM 4 KB

Velocidad de reloj 16 MHz

Esquemático.

A continuación se expone el esquemático de Arduino Mega 2560, para su mejor vista se ha dividido en dos imágenes.





Alimentación

El Arduino Mega puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.

Las fuentes de alimentación externas (no-USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería puede conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER)

La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable, si se usan más de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.

Los pines de alimentación son los siguientes:

VIN. La entrada de voltaje a la placa Arduino cuando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentado a través de la conexión de 2.1mm, acceder a ella a través de este pin.

5V. La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB o otra fuente estabilizada de 5V.

3V3. Una fuente de voltaje a 3.3 voltios generada en el chip FTDI integrado en la placa. La corriente máxima soportada 50mA.

GND. Pines de toma de tierra.

Memoria

El ATmega1280 tiene 128KB de memoria flash para almacenar código (4KB son usados para el arranque del sistema (bootloader)). El ATmega1280 tiene 8 KB de memoria SRAM. El ATmega1280 tiene 4KB de EEPROM, que puede a la cual se puede acceder para leer o escribir con la [Reference/EEPROM |librería EEPROM]].

Entradas y Salidas

Cada uno de los 54 pines digitales en el Duemilanove pueden utilizarse como entradas o como salidas usando las funciones `pinMode()`, `digitalWrite()`, y `digitalRead()` . Las E/S operan a 5

voltios. Cada pin puede proporcionar o recibir una intensidad máxima de 40mA y tiene una resistencia interna (desconectada por defecto) de 20-50kOhms. Además, algunos pines tienen funciones especializadas:

Serie: 0 (RX) y 1 (TX), Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX). Usado para recibir (RX) transmitir (TX) datos a través de puerto serie TTL. Los pines Serie: 0 (RX) y 1 (TX) están conectados a los pines correspondientes del chip FTDI USB-to-TTL.

Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2). Estos pines se pueden configurar para lanzar una interrupción en un valor LOW(0V), en flancos de subida o bajada (cambio de LOW a HIGH(5V) o viceversa), o en cambios de valor. Ver la función `attachInterrupt()` para más detalles.

PWM: de 0 a 13. Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a través de la función `analogWrite()`.

SPI: 50 (SS), 51 (MOSI), 52 (MISO), 53 (SCK). Estos pines proporcionan comunicación SPI, que a pesar de que el hardware la proporcione actualmente no está incluido en el lenguaje Arduino.

LED: 13. Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

El Mega tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide de tierra a 5 voltios, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función `analogReference()`. Además algunos pines tienen funciones especializadas:

I2C: 20 (SDA) y 21 (SCL). Soporte del protocolo de comunicaciones I2C (TWI) usando la librería Wire.

Hay otros pines en la placa:

AREF. Voltaje de referencia para las entradas analógicas. Usado por `analogReference()`.

Reset. Suministrar un valor LOW(0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los shields que no dejan acceso a este botón en la placa.

Comunicaciones

EL Arduino Mega facilita en varios aspectos la comunicación con el ordenador, otro Arduino u otros microcontroladores. El ATmega1280 proporciona cuatro puertos de comunicación vía serie UART TTL (5V). Un chip FTDI232RL integrado en la placa canaliza esta comunicación serie a través del USB y los drivers FTDI (incluidos en el software de Arduino) proporcionan un puerto serie virtual en el ordenador. El software incluye un monitor de puerto serie que permite enviar y recibir información textual de la placa Arduino. Los LEDs RX y TX de la placa parpadearán cuando se detecte comunicación transmitida través del chip FTDI y la conexión USB (no parpadearán si se usa la comunicación serie a través de los pines 0 y 1).

La librería SoftwareSerial permite comunicación serie por cualquier par de pines digitales de Mega.

El ATmega1280 también soportan la comunicación I2C (TWI) y SPI . El software de Arduino incluye una librería Wire para simplificar el uso el bus I2C, ver The la documentación para más detalles. Para el uso de la comunicación SPI, mira en la hoja de especificaciones (datasheet) del ATmega1280.

Programación

El Arduino Mega se puede programar con el software Arduino (descargar). Para más detalles mirar referencia y tutoriales.

El ATmega1280 en el Arduino Mega viene precargado con un gestor de arranque (bootloader) que permite cargar nuevo código sin necesidad de

un programador por hardware externo. Se comunica utilizando el protocolo STK500original(referencia, archivo de cabecera C).

También te puedes saltar el gestor de arranque y programar directamente el microcontrolador a través del puerto ISCP (In Circuit Serial Programming);para más detalles ver estas instrucciones.

Reinicio Automático por Software

En vez de necesitar reiniciar presionando físicamente el botón de reset antes de cargar, el Arduino Mega está diseñado de manera que es posible reiniciar por software desde el ordenador donde esté conectado. Una de las líneas de control de flujo (DTR) del FT232RL está conectada a la línea de reinicio del ATmega1280 a través de un condensador de 100 nano faradios. Cuando la línea se pone a LOW (0V), la línea de reinicio también se pone a LOW el tiempo suficiente para reiniciar el chip. El software de Arduino utiliza esta característica para permitir cargar los sketches con solo apretar un botón del entorno. Dado que el gestor de arranque tiene un lapso de tiempo para ello, la activación del DTR y la carga del sketch se coordinan perfectamente.

Esta configuración tiene otras implicaciones. Cuando el Mega se conecta a un ordenador con Mac OS X o Linux, esto reinicia la placa cada vez que se realiza una conexión desde el software (vía USB). El medio segundo aproximadamente posterior, el gestor de arranque se está ejecutando. A pesar de estar programado para ignorar datos mal formateados (ej. cualquier cosa que la carga de un programa nuevo) intercepta los primeros bytes que se envían a la placa justo después de que se abra la conexión. Si un sketch ejecutándose en la placa recibe algún tipo de configuración inicial u otro tipo de información al inicio del programa, asegúrate que el software con el cual se comunica espera un segundo después de abrir la conexión antes de enviar los datos.

El Mega contiene una pista que puede ser cortada para deshabilitar el auto-reset. Las terminaciones a cada lado pueden ser soldadas entre ellas para rehabilitarlo. Están etiquetadas con "RESET-EN". También podéis deshabilitar el auto-reset conectando una resistencia de 110 ohms desde el pin 5V al pin de reset; mirar este hilo del foro para más detalles.

Protección contra sobretensiones en USB

El Arduino Mega tiene un multifusible reinicializable que protege la conexión USB de tu ordenador de cortocircuitos y sobretensiones. A parte que la mayoría de ordenadores proporcionan su propia protección interna, el fusible proporciona una capa extra de protección. Si más de 500mA son detectados en el puerto USB, el fusible automáticamente corta la conexión hasta que el cortocircuito o la sobretensión desaparecen.

Características Físicas y Compatibilidad de Shields

La longitud y amplitud máxima de la placa Duemilanove es de 4 y 2.1 pulgadas respectivamente, con el conector USB y la conexión de alimentación sobresaliendo de estas dimensiones. Tres agujeros para fijación con tornillos permiten colocar la placa en superficies y cajas. Ten en cuenta que la distancia entre los pines digitales 7 y 8 es 160 mil (0,16"), no es múltiple de la separación de 100 mil entre los otros pines.

El Mega está diseñado para ser compatible con la mayoría de shields diseñados para el Diecimila o Duemilanove. Los pines digitales de 0 a 23 (y los pines AREF y GND adyacentes), las entradas analógicas de 0 a 5, los conectores de alimentación y los conectores ICPS están todos ubicados en posiciones equivalentes. Además el puerto serie principal está ubicado en los mismos pines (0 y 1), así como las interrupciones 0 y 1 (pines 2 y 3 respectivamente). SPI está disponible en los conectores ICSP tanto en el mega como en el Duemilanove/Diecimila. Atención, los pines I2C no están ubicados en la misma posición en el Mega (20 y 21) que en el Duemilanove/Diecimila (entradas analógicas 4 y 5).

Módulos necesarios anexados a Arduino

Modulo Puente H L298



El módulo puente H L298N es una tarjeta para el control de motores de corriente directa, motores a pasos, solenoides y en general cualquier otra carga inductiva. La tarjeta está construida en torno al circuito integrado L298N, el cual dispone en su interior de 2 puentes H independientes con capacidad de conducir 2 amperios constantes o 4 amperios en picos no repetitivos. La tarjeta expone las conexiones hacia el motor a través de bloques de terminales (clemas), mientras que las entradas de control y habilitación del puente H se exponen a través de headers macho estándar para facilitar todas las conexiones.

Esta tarjeta es ideal para controlar motores en pequeños robots como seguidores de líneas, zumbos, robots de laberinto, etc. El L298N también es una excelente opción para manejar motores a pasos bipolares.

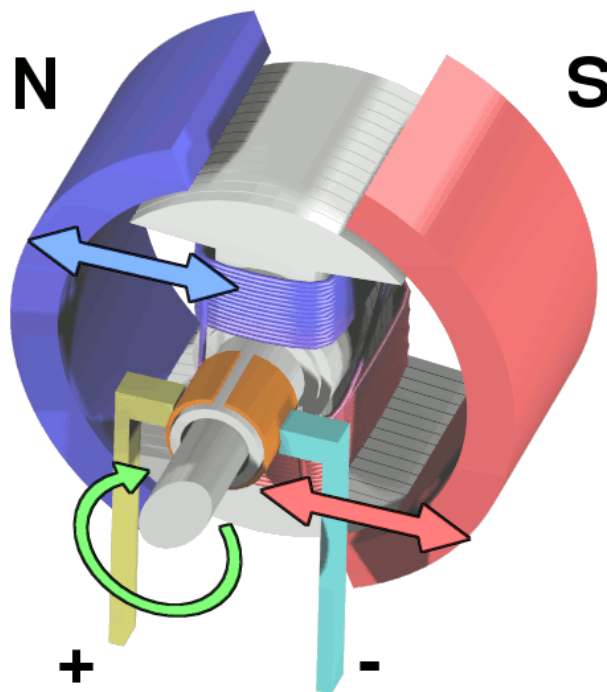
Recomendamos utilizar cables tipo dupont para conectar las señales de control.

Características del Módulo puente H L298N:

- Circuito Integrado principal: L298N
- Corriente pico de operación: 4 Amperios
- Corriente constante de operación: 2 Amperios
- Bajo voltaje de saturación en los transistores de salida
- Corte de operación por sobrecalentamiento
- Voltaje de alimentación de motores de hasta 46 volts
- Excelente inmunidad al ruido
- Ideal para controlar motores en aplicaciones de robótica

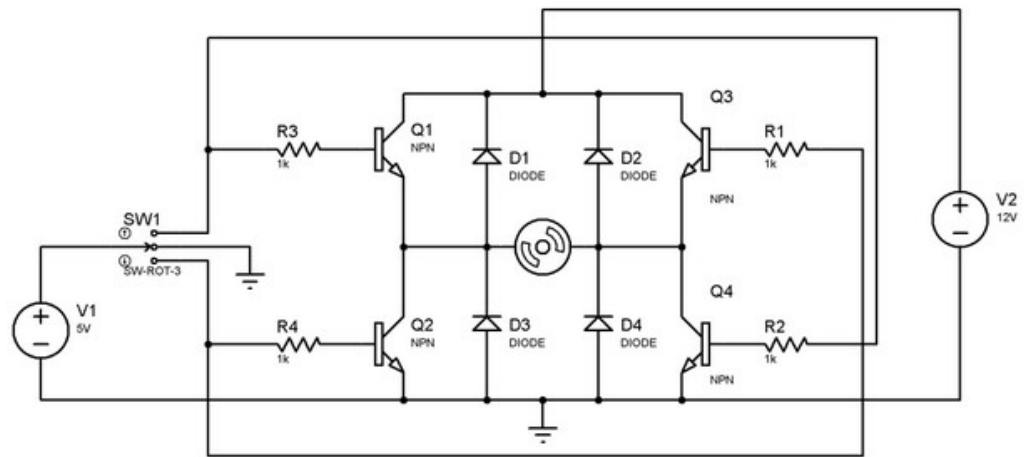
El puente H le proporciona a Arduino el manejo de los motores, en este proyecto solamente precisamos cambiar el giro con el que se mueve el motor, alterando su polaridad pero creemos necesario mencionar que también podremos controlar su velocidad si lo quisiéramos. Explicaremos un poco cómo logra el puente H cambiar el giro de los motores que tiene conectados:

Tenemos un motor de corriente continua como indica la imagen:



Para invertir el sentido de giro de un motor hay que cambiarle la polaridad, es decir, cambiar el sentido con el que la corriente pasa a través del motor.

En electrónica esto normalmente se logra con una configuración llamada puente H, la cual lo indicamos en el siguiente esquema:

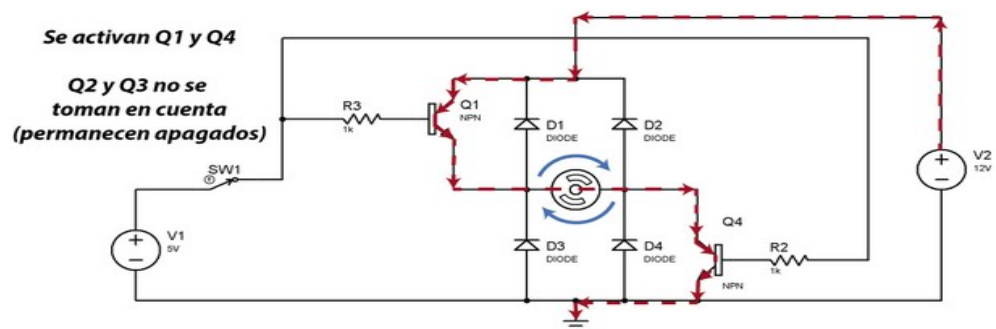


En el interruptor SW1 se decide si el motor gira a la izquierda, a la derecha o si se detiene.

Cuando el interruptor está en la posición del centro, no hay voltaje aplicado a los transistores por lo que permanecen en estado de corte.

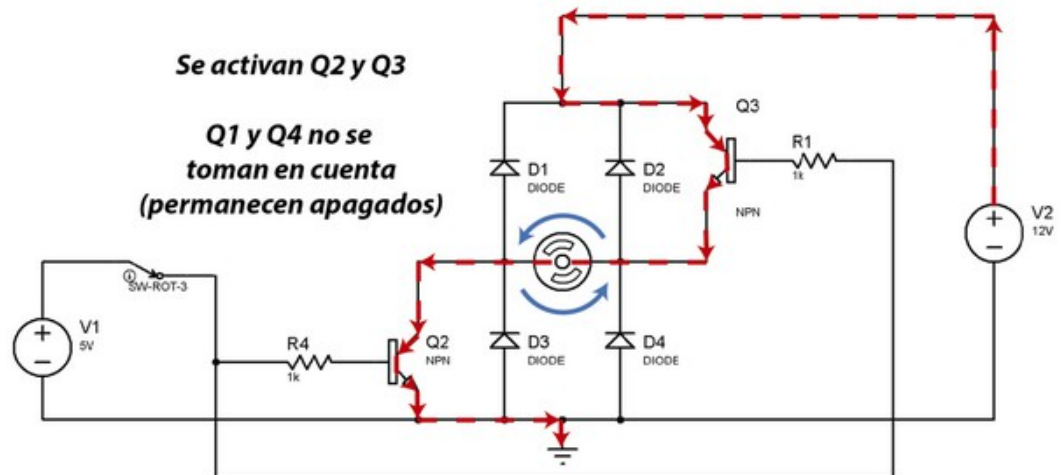
Los transistores se utilizan como interruptores y como dispositivos de control.

Cuando se coloca el interruptor en la primera posición (la superior), el comportamiento de la corriente es el siguiente:



La corriente fluye a través de Q1 y Q4. El motor gira en sentido horario.

Si se cambia de posición el interruptor entonces el motor gira en sentido anti horario.



Para activar los transistores se requieren tensiones muy bajas, lo que hace que podamos reemplazar el interruptor por Arduino y es por esto que nuestro puente H posee el integrado L298D para la comunicación con dicha placa.

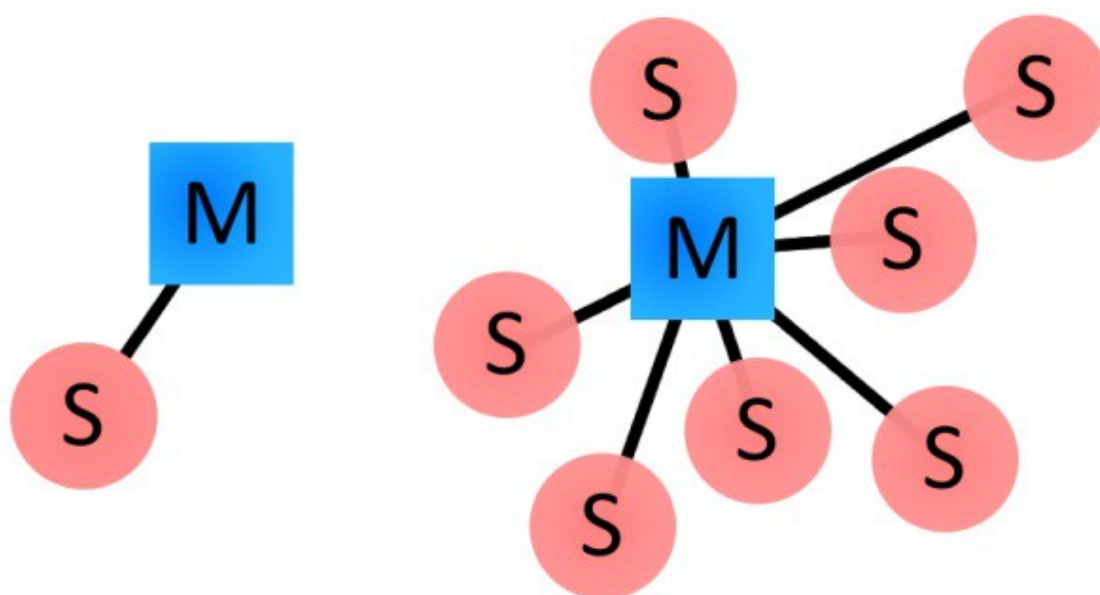
Comunicación vía Bluetooth

Bluetooth es una especificación industrial para Redes Inalámbricas (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 Ghz. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles.
- Eliminar los cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Los dispositivos Bluetooth pueden actuar como Masters o como Slaves (Amos o esclavos).

La diferencia es que un Bluetooth Slave solo puede conectarse a un master y a nadie más, en cambio un master Bluetooth, puede conectarse a varios Slaves o permitir que ellos se conecten y recibir y solicitar información de todos ellos, arbitrando las transferencias de información (Hasta un máximo de 7 Slaves)



Cada uno de los dispositivos que se identifican vía BlueTooth presentan una dirección única de 48 bits y además un nombre de dispositivo que nos sirva para identificarlo cómodamente a los humanos. Por eso cuando configuras tu móvil puedes especificar un nombre propio que será el que mostrarás a los demás cuando busquen tu teléfono en la inmediaciones.

La dirección propia también se puede identificar pero lógicamente, es un poco menos cómoda y tiene menos utilidad. Tampoco es raro establecer un protocolo IP sobre transporte BlueTooth, con lo que además de su identificación interna BlueTooth (Equivalente al MAC Ethernet) dispondrá de una dirección IP para conectarse a Internet.

Por eso puedes conectarte vía Bluetooth a tu PC, por ejemplo, y a través de él conectarte a internet.

Así pues un nodo BlueTooth puede ser Master o Slave y dispone de una dirección única, así como de un nombre para identificarse y muy habitualmente también incluye un PIN de conexión o número de identificación que debe teclearse para ganar acceso al mismo.

Como el Bluetooth lo desarrolló Nokia para conectar teléfonos móviles, a otros dispositivos como auriculares, micrófonos o conexiones al audio del coche, existe un procedimiento definido que se llama Pairing (o emparejamiento) que vincula a dos dispositivos Bluetooth.

Cuando vinculas dos dispositivos BT, se inicia un proceso en el que ellos se identifican por nombre y dirección interna y se solicitan la clave PIN para autorizar la conexión.

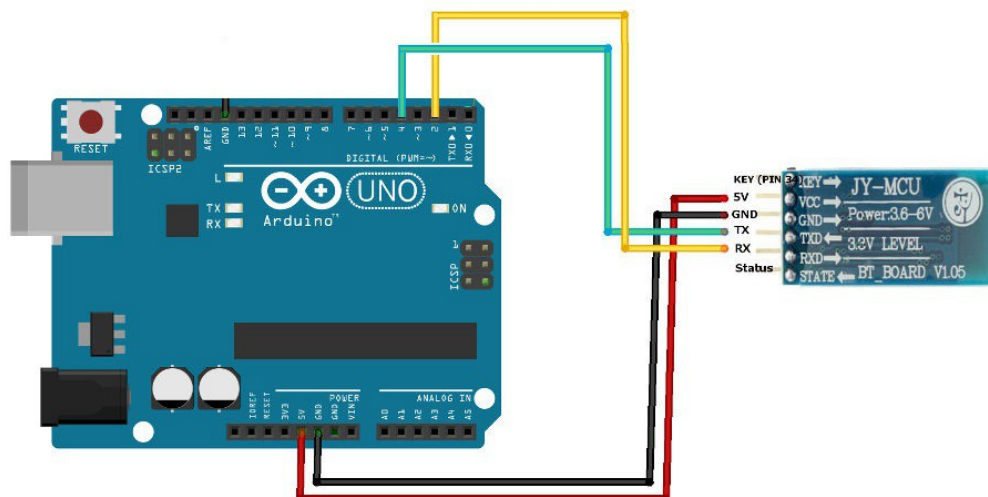
Si el emparejamiento se realiza con éxito, ambos nodos suelen guardar la identificación del otro y cuando se encuentran cerca se vuelven a vincular sin necesidad de intervención manual. Por eso el CD de tu coche reconoce el móvil de tu bolsillo en cuanto te subes y puedes reproducir la música que tienes en tu Smartphone.

Existen varios dispositivos que permiten la comunicación Bluetooth en Arduino, elegimos utilizar el módulo HC - 06 el cual es un módulo que funciona como esclavo y se pueden conseguir de distintas formas, una de ellas es con 4 pines.



La conexión con Arduino es fácil, posee 4 pines los cuales son:

GND, tensión, TX y RX. Estos últimos van a los pines digitales TX y RX de la placa arduino pero en sentido inverso, es decir TX del bluetooth se conecta con RX de la placa arduino, anexamos una imagen meramente ilustrativa:

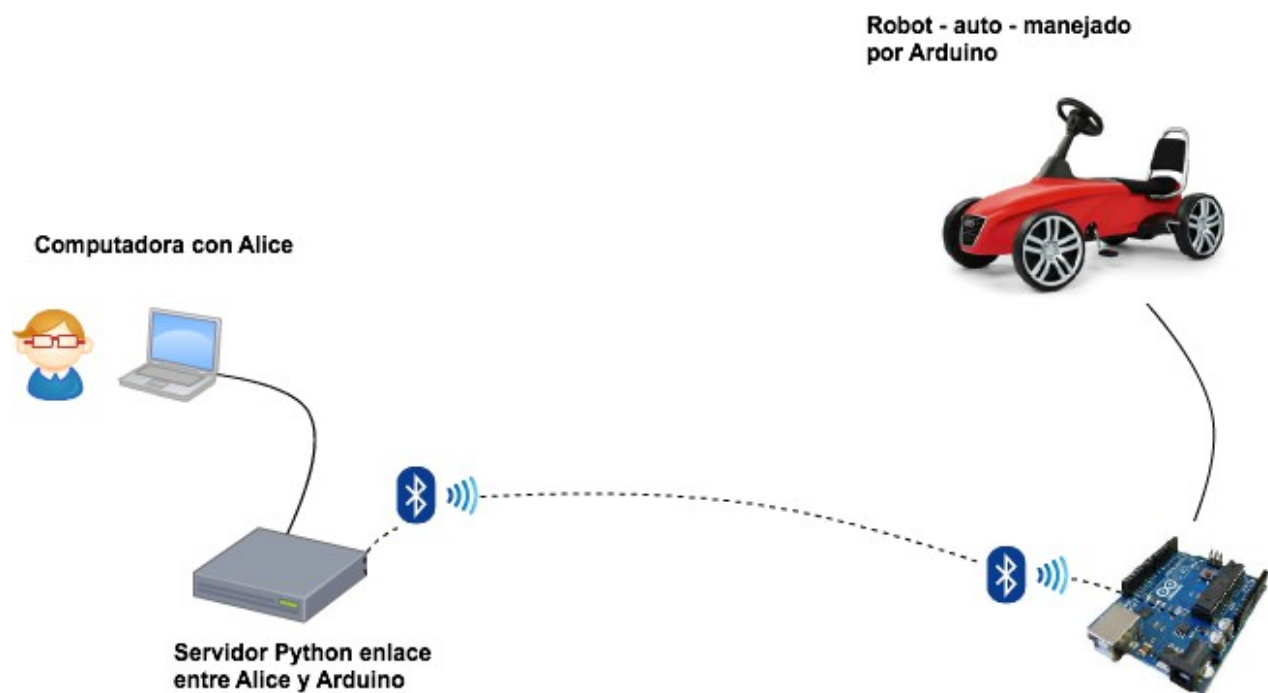


La comunicación se realiza a través del soporte Serial que indicamos anteriormente de Arduino y veremos más adelante como se programa.

CAPÍTULO VII

DESARROLLO DE INGENIERÍA

DIAGRAMA DE BLOQUES



Descripción

Este diagrama simboliza el alcance del proyecto Ana. Si bien la comunicación es en tiempo real el flujo de datos tiene un sentido y es el siguiente.

Cuando el usuario, en este caso simbolizado por el nene con lentes utiliza Alice en su computadora, realiza desde este software un movimiento, Alice procede a informarle al servidor que hubo un cambio que debe informarle a Arduino. EL concepto servidor que abordaremos más adelante fue logrado a través de socket y puede alojarse en la misma computadora donde se encuentra Alice o en cualquier lugar del mundo, accesible por internet.

Siguiendo con el flujo de comunicación cuando el servidor recibe la instrucción, comprueba si dicha instrucción está dentro de sus movimientos válidos y realiza la escritura del movimiento a través del dispositivo Bluetooth, cabe aclarar que cuando el servidor se pone en contacto con Alice, por mas que no hayan movimientos todavía, el ya busca sus dispositivos recordados y si Arduino está encendido se enlaza para no perder tiempo en la comunicación.

El paso final es cuando el movimiento llegó a la placa Arduino, la cual al recibir el movimiento, también chequea si está dentro de sus movimientos aceptados y le envía la instrucción al auto para que se mueva. En este caso, el encargado de mover el auto es el puente H, por lo que activara, cambiara el giro o desactivara los motores conectados al puente.

Como el objetivo de este trabajo no era realizar un robot completo, hemos hecho dos movimientos válidos y uno por defecto, el cual es quedarse quieto.

La lista de movimientos es:

1. Detenido.
2. Hacia adelante.
3. Hacia atrás.

Siguiendo con la lógica antes expuesta repasamos la interacción a nivel movimientos para que quede claro cuál es el flujo de trabajo entre Alice y el Robot.

Cuando un usuario ingresa a Alice puede optar por ir para adelante o hacia atrás, manteniendo la flecha del teclado de navegación, si la deja presionada, el robot se sigue moviendo y cuando la suelta se detiene.

Por defecto el robot está quieto, mientras el usuario presiona para adelante o para atrás el robot se mueve en esos sentido y cuando se deja de presionar se detiene. La comunicación en este caso está planteada desde la computadora al Robot y es meramente por el alcance de este trabajo pero podría ser bidireccional.

Se consiguió esta interacción gracias a Alice que como dijimos anteriormente es un lenguaje de programación educativo libre y orientado a objetos, con un entorno de desarrollo integrado, que nos brinda una gran variedad de herramientas destinadas a poder incentivar al alumno en el mundo del desarrollo orientado a objetos. Veremos un poco la historia de Alice, como nos brinda estas herramientas y sobre todo veremos cómo utilizamos esta herramienta para poder amoldarse a la robótica, y especialmente al proyecto Ana.

Se investigó todas las funcionalidades de Alice y se logró la comunicación de Alice hacia nuestro servidor a través de Scripting que viene incorporado en Alice en sus versiones 2.2, 2.3 y 2.4.

Este Script dentro de Alice es quien interactúa con nuestro servidor, por lo que también explicaremos que es un socket y como se programo en Python para recibir conexiones desde distintos Robots.

Una vez que quede claro la comunicación entre Alice, se explicará detalladamente cómo se creó el robot, para que tenga autonomía y esté esperando recibir instrucciones a través de Bluetooth y como se programo para cumplir estas funciones.

Por último haremos una demostración con todos los actores involucrados, por lo que para este caso simularemos un auto que tenga los movimientos hacia adelante y hacia atrás. A su vez encenderemos nuestro robot y este a su vez encenderá el Bluetooth para esperar conexiones e instrucciones desde el servidor.

Alice se comunicara con el servidor y le enviará instrucciones a nuestro robot, veremos una comunicación en tiempo real. Es decir, el auto en la pantalla de nuestra computadora avanzara y el robot también, será un espejo de movimientos pero la idea es demostrar que el tiempo real a través de Alice es posible.

CAPITULO VIII

ALICE

Introducción

Al momento de elegir Alice como software educativo al cual le anexamos un robot en Arduino, estuvo el dilema de como lograr esta comunicación ya que el software no fue creado para interactuar con otros dispositivos.

Confiamos en que al ser Open Source iba a tener una buena documentación y en el caso de ser necesario crear una nueva versión a partir del código fuente y anexar nuestro modulo de comunicación hacia Arduino.

Para familiarizarnos con este software, crearemos un proyecto de cero mostrando alguna de sus funcionalidades y luego demostraremos como se logró incorporar el modulo de comunicacion.

Instalación de ALICE 2.4

A continuación detallaremos los pasos que debemos seguir para instalar en la Notebook el software Alice.

El software Alice como mencionamos anteriormente es una aplicación libre y gratuita, para bajarla debemos acceder a la página de Alice y descargar una de las versiones de Alice, para los sistemas operativos más conocidos, a saber: Linux, Windows y en este caso Mac OS.

Las pruebas se realizaron sobre la versión de Alice 2.4, actualmente en la página a la hora de terminar el proyecto, ya disponible la versión de Alice 3.1, queda a criterio del alumno o docente qué versión utilizar, en este documento hacemos énfasis en la versión de Alice 2.4. Y el siguiente tutorial de instalación se hace sobre la versión de Alice 2.4.

Requerimientos

La herramienta Alice está diseñada en Java, por lo tanto necesitamos tener alguna versión de java instalada en la Notebook.

En este documento no haremos hincapié en la instalación de Java dado que ya viene como software pre instalado en las netbook, pero si nos encontramos con alguna netbook que no tenga el software solo debemos entrar a la url de Java y descargar la última versión disponible dependiendo que sistema operativo tenemos instalado.

Url de descarga de java: “<http://java.com/en/download/>”

El instalador de Alice se baja por lo general comprimido con la extensión .zip, por lo tanto necesitamos alguna aplicación para descomprimirlo, podremos utilizar compresor 7-Zip, el cual es gratuito, por si nos lo tenemos instalado a continuación dejamos la url de descarga.

URL de descarga de 7-Zip: “<http://www.7-zip.org/>”

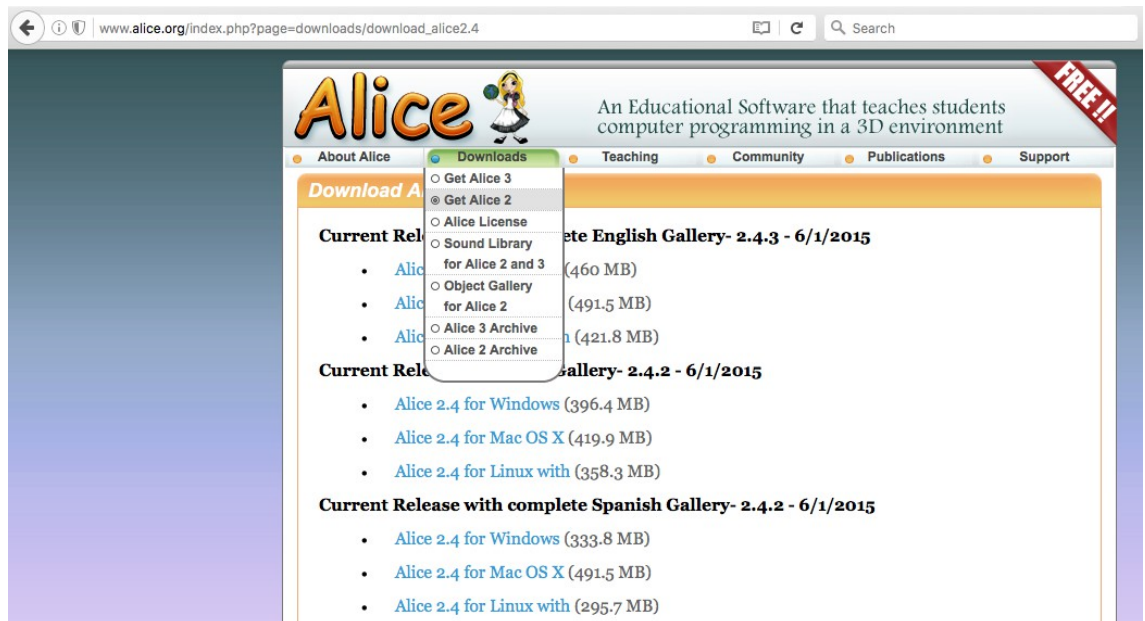
Descarga Alice

Ingresamos al siguiente enlace:

http://www.alice.org/index.php?page=downloads/download_alice2.4

Como podremos observar, veremos varias versiones, para este trabajo hemos utilizado la versión 2 de Alice y más precisamente la 2.4 por cuestiones de compatibilidad al momento de comenzar a trabajar con Scripting, lo cual veremos más adelante.

Al ingresar al sitio, vemos:



Instalación de Alice 2.4

Al descomprimir la descarga .zip obtendremos una imagen .dmg, la cual deberemos arrastrar a Aplicaciones en Mac OS.

Anexamos también la instalación en Windows:

1. Debemos descomprimir la carpeta ZIP.
2. Luego debemos acceder a dicha carpeta y buscar el archivo llamado "SlowAndSteadyAlice.exe" y al hacer doble clic, se abrirá el programa, debemos asegurarnos de tener Java instalado.

Si todo fue bien veremos una pantalla como la siguiente al inicializarse:



El paso siguiente al iniciar es crear un proyecto para que lo vamos a tener que basarnos en algunas condiciones, por lo que lo explicaremos en el siguiente apartado, el cual es creacion de una aplicación en Alice.

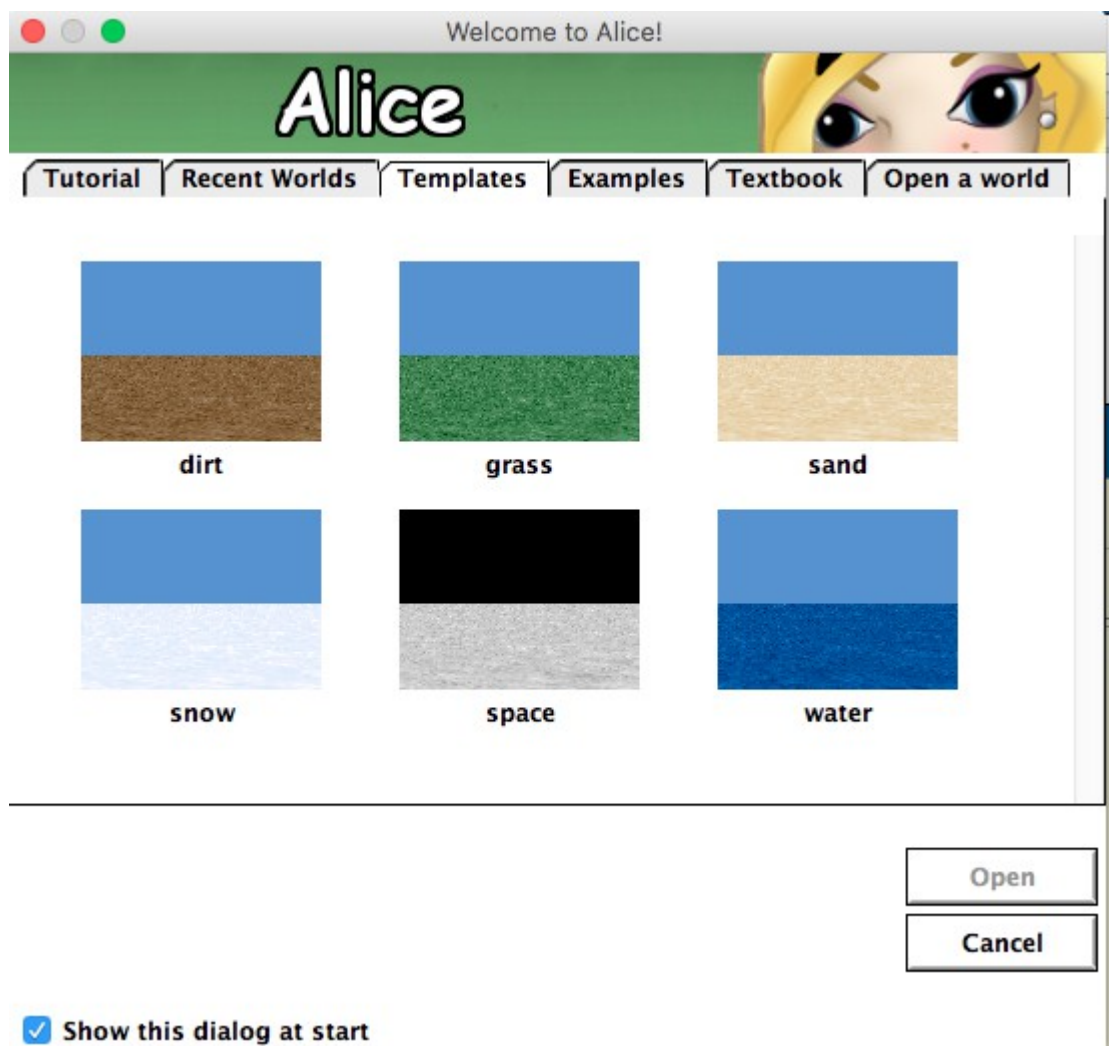
Crear una aplicación en Alice

A modo introductorio al software Alice vamos a realizar una aplicación básica donde crearemos un escenario y añadiremos unos objetos para que el lector se vaya familiarizando con la interfaz que nos brinda Alice.

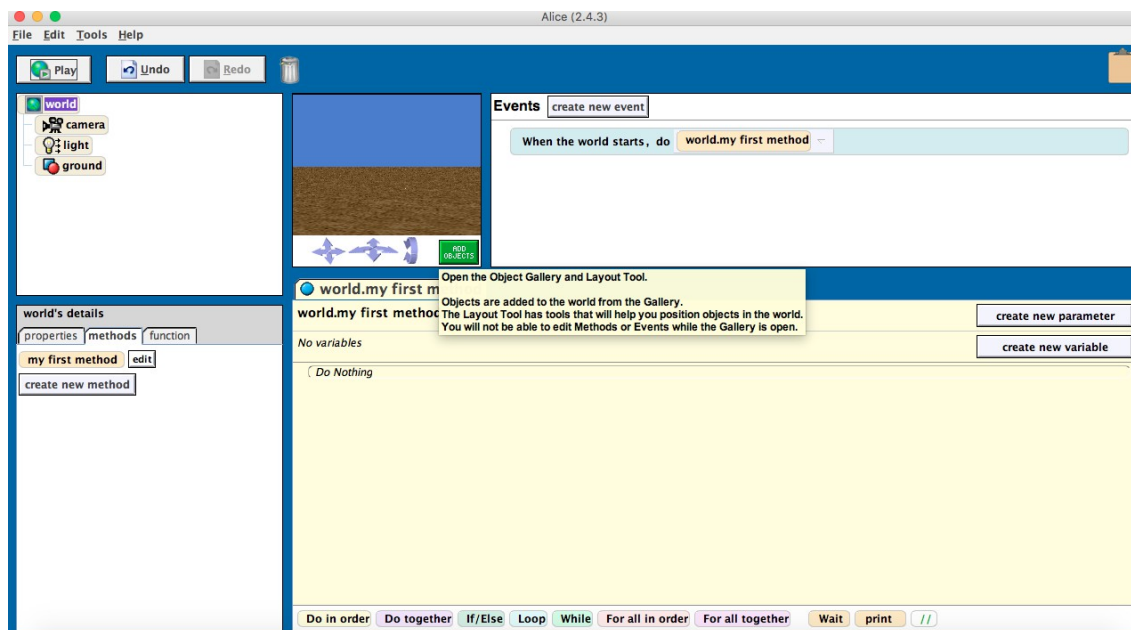
Al momento de abrir la aplicación por primera vez, Alice nos propondrá configurar nuestro nuevo proyecto, debemos tener en cuenta algunas cuestiones ya que una vez creados no se puede cambiar.

Escenarios

Cuando iniciamos Alice nos pregunta qué tipo de escenario deseamos para nuestra aplicación, la siguiente imagen ilustra este caso:

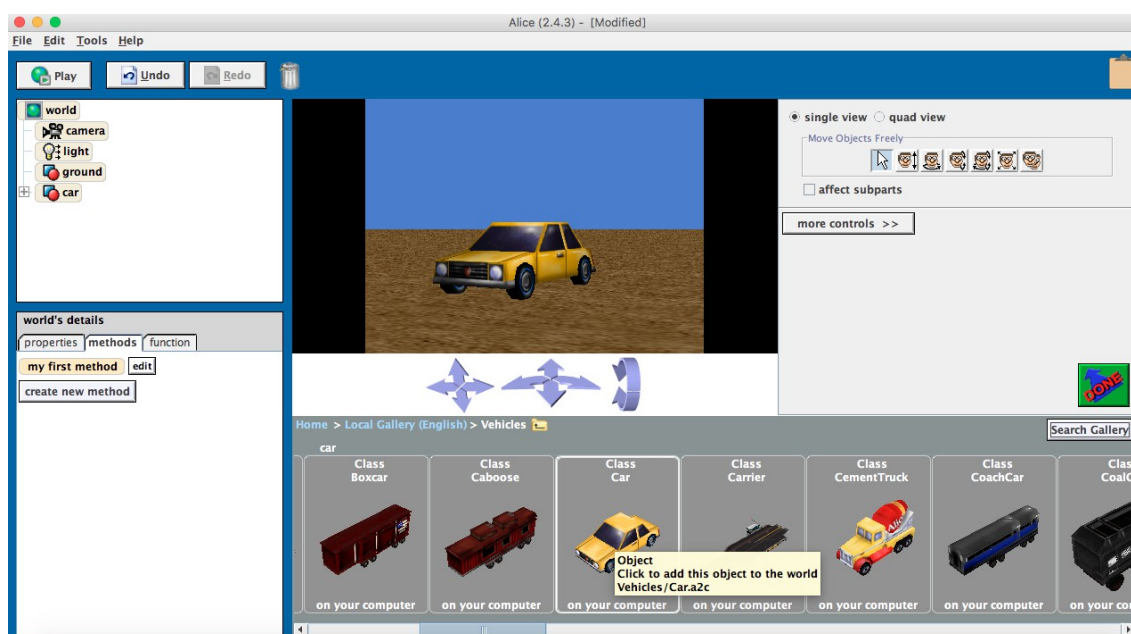


Una vez seleccionado el escenario tendremos una plataforma de trabajo como la siguiente:

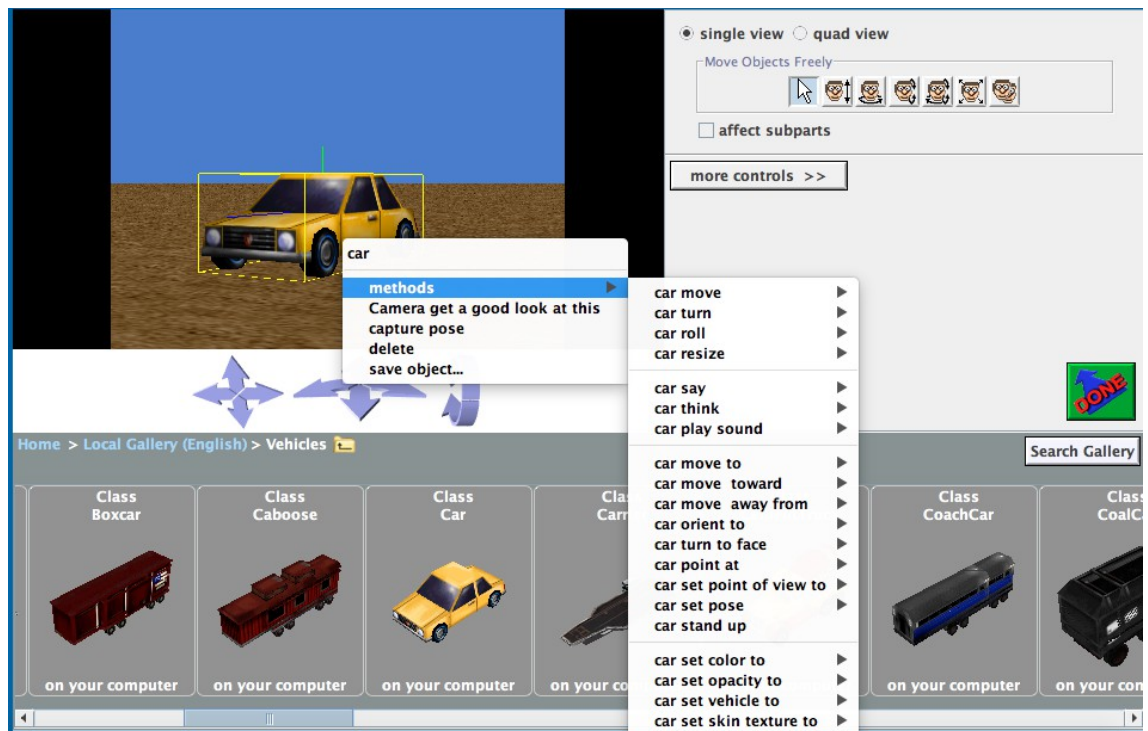


Objetos

Cuando hacemos clic en el botón Add Object se nos abre un listado de categorías donde cada una tiene sus objetos listos para ser añadidos a la aplicación:



En este caso hemos agregado un auto amarillo a nuestro escenario, si realizamos clic sobre dicho objeto veremos que tenemos un listado de movimientos en la categoría llamada methods.



Una vez que dejamos nuestro objeto como quisiéramos que aparezca desde el momento cero, presionamos en el botón DONE o HECHO, que podemos ver situado a la derecha.

Ya tenemos nuestro escenario con un objeto creado listo para realizar lo que nosotros deseemos.

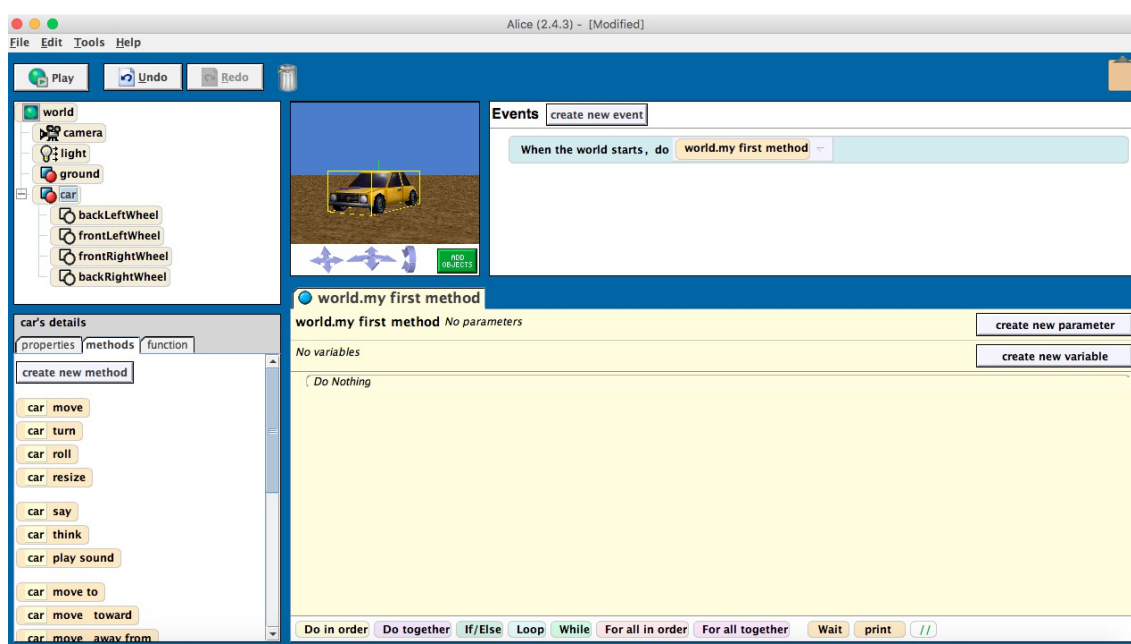
Cabe aclarar que los objetos pueden ser agregados o removidos siempre. Podremos cambiar las cámaras que enfoque a un determinado objeto, es decir, la pantalla principal apunte a un objeto.

La idea de Alice es que los objetos interactúan entre sí o a través de movimientos que ingresa el usuario. Podremos crear objetos que no requieran movimientos del exterior, una especie de video o animación, puede tener un inicio, un desenlace y un fin en base a determinadas condiciones que creamos en nuestro programa, por ejemplo decirle a un Auto que vaya y si encuentra un disco Pare, se detenga y termine su ejecución y si no lo encuentra, que circule en distintas direcciones por todo el escenario.

Y también podremos crear objetos que esperen instrucciones de un usuario, un juego donde el objeto realiza acciones en base a los movimientos que envía el usuario.

Movimientos e interacción

En el escenario ya listo para trabajar, obtendremos una pantalla como la siguiente:



En el panel de la derecha tendremos los atributos (properties) de nuestro auto, es decir, podremos setear colores, opacidad, posición, etc.

También tendremos métodos (*methods*) ya creados para este objeto auto.

Y nos queda la pestaña function, la cual son los métodos personalizados que nosotros crearemos para nuestro objeto.

Este es el objetivo de Alice, brindarnos conceptos de orientación a objetos de una manera distinta para que cualquier persona que no sepa de programación pueda entenderlos e incorporarlos a su conocimiento.

Siguiendo con la presentación del panel superior vemos que tenemos a la derecha del escenario y el objeto una pantalla. Ahí se encuentran los eventos, porque a pesar que es orientada a objetos la programación en

Alice, también acepta el paradigma orientado a eventos. Por ejemplo podemos configurar que cuando el mundo comience, el programa haga algo.

Esto hace referencia a cuando el usuario corre el programa. Dentro de ese listado de eventos tenemos varios que pueden ayudarnos a manejar el flujo de nuestro programa. Por ejemplo podemos decir, mientras la variable X sea igual a tal valor, hacer esto, podremos hacer acciones cuando la tecla Y se presiona, etc.

El objetivo de esta introducción no es que crear un manual de Alice, si no brindar un paneo general de la herramienta para que algunos conceptos que posteriormente vamos a exponer no suenen extraños.

Scripting

Para lograr el objetivo del proyecto Ana se investigó todas las funcionalidades de Alice y ver donde podíamos realizar la comunicación en tiempo real con nuestro servidor.

En un primer momento buscamos encontrar algún tipo de log que haga la aplicación para delimitarlo con algún software externo y enviarlos a nuestro robot. Si bien posee un log este no enviaba información válida del movimiento, si no información propiamente de java, acerca de sus errores o su éxito en la ejecución.

Luego pensamos en obtener el código fuente de Alice, ya que al ser Open Source es público, y realizarle una iteración más agregándole nuestro módulo de comunicación.

Esta opción sigue siendo válida y ahora les contaremos porque se descarto.

Cuando conseguimos el código de Alice, se hizo muy difícil conseguir a alguien que esté familiarizado con la versión 2 de Alice, la versión 3 en ese momento estaba en beta por lo que decidimos realizarlo en la 2. Al no encontrar alguien que nos de soporte de como compilar la herramienta, comenzamos a probar y se iba avanzando, despacio pero era un avance.

Creamos un repositorio en git y realizamos un fork de dicho repositorio, logramos compilar algunos módulos pero teníamos muchos métodos deprecados y la tarea se estaba haciendo difícil. Al momento de encontrar a alguien que nos ayude, estuvimos en contacto con personas del foro de Alice, lo cual nos ayudó mucho a la solución final.

El foro esta bastante atractivo ya que tiene una buena comunidad, pueden acceder en:

<http://www.alice.org/community/>

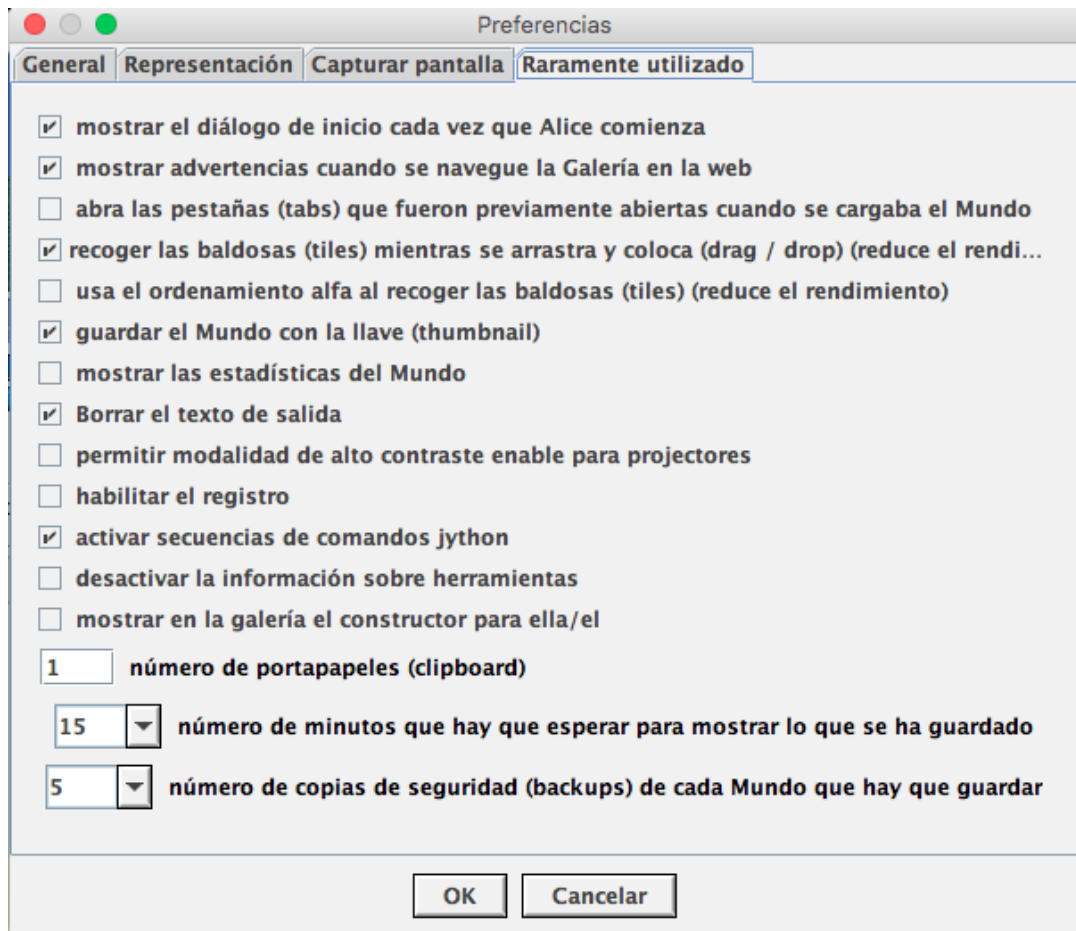
Nos mencionaron que Alice poseía Scripting en Jython, este lenguaje es Python que corre arriba de la virtual machine de Java, algo así como Grails, que es Rails arriba de la VM de Java. Sin duda esto no era la solución final pero había que ver de qué se trataba ese Scripting.

Pudimos lograr correrlo y realizar algunas funciones, si bien la versión de Jython está desactualizada, logramos crear un Socket en localhost donde esta corriendose Alice, la dirección localhost fue arbitraria, hubiésemos podido realizar una comunicación con un servidor tranquilamente.

Entonces la idea general a esta solución fue la siguiente:

Cuando un objeto realiza un movimiento en Alice, también escribe en nuestro socket un valor de referencia para que el servidor lo recoja y lo reenvíe a nuestro robot. Estamos hablando de una comunicación de un carácter por movimiento por lo que el retardo de esta comunicación es de milisegundos. A continuación activaremos Scripting en Alice

Debemos dirigirnos a **Modificar | Preferencias** y hacemos clic en *activar secuencias de comandos Jython*, le damos clic en ok y esto nos pedirá que reiniciemos Alice:



Una vez reiniciado veremos que ahora arriba de nuestro escenario habrá un prompt para ingresar scripting:



Esto nos brindara una consola para que ingresemos líneas de script en lenguaje Jython, también podremos crear un script con métodos donde se

podrá llamar desde cualquier parte de Alice, este es el caso nuestro y lo debemos editar en el Mundo que estamos trabajando, para esto vamos al panel izquierdo y en Mundo hacemos clic derecho:

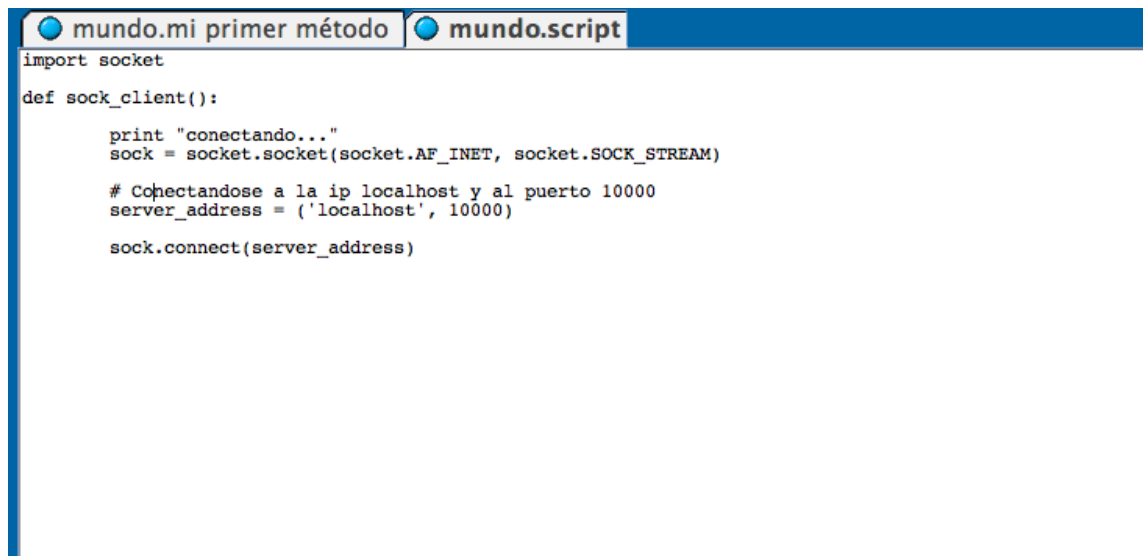


En modificar el guión (script) nos abrirá un editor de texto llamado **mundo.script**, como podemos apreciar abajo de nuestro mundo, donde podremos crear funciones y ser invocadas a lo largo de nuestro proyecto.

Crearemos un socket cliente que se conecte a localhost y en el puerto 10000, la ip y el puerto son elecciones nuestras, podremos conectarnos a un servidor en cualquier ip, siempre y cuando acepte conexiones del exterior en ese puerto. Los puertos por convención están reservados por el sistema operativo en el rango de 0 a 1023 por lo que nosotros podríamos utilizar cualquier por arriba de 1023 hasta el número de puerto 65535, en nuestro ejemplo escogimos el número 10000.

Para que nuestro socket cliente funcione necesita que nuestro servidor esté escuchando en la misma ip (localhost o 127.0.0.1) y en el puerto 10000, lo cual lo veremos en el capítulo posterior.

Entonces una vez que editamos el script ya estamos listos para empezar a intercambiar información con nuestro servidor:



```
import socket

def sock_client():

    print "conectando..."
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Conectandose a la ip localhost y al puerto 10000
    server_address = ('localhost', 10000)

    sock.connect(server_address)
```

Al instanciar el objeto **socket**, obtendremos nuestra instancia **sock** el cual tendrá métodos, por ejemplo leer y escribir al socket, esto es el principio básico de comunicación que nos provee el socket. Para nuestro objetivo, Alice solamente enviará información al servidor por lo que utilizaremos el método **.send** para enviar información.

Solo nos falta incorporar nuestro script en Alice, es decir crear una instancia, para esto en el panel principal, en la parte inferior vemos que tenemos un tag llamado Guión (script).



Vemos que al agregarlo debemos agregar el nombre de nuestro método del script, donde también podremos enviarle información, en nuestro proyecto que veremos más adelante, le enviaremos información al servidor para que sepa qué movimiento debe enviar al robot Arduino.

Es decir, que nuestro script será ejecutado cada vez que se realice un movimiento en Alice, si el auto se detiene enviaremos una señal de PARE, si avanza o retrocede también.

En el próximo capítulo crearemos un servidor para que intercambie datos con Alice, en este caso haremos una comunicación bidireccional como ejemplo y luego al finalizar haremos la comunicación en el sentido que fue propuesto para el Proyecto Ana.

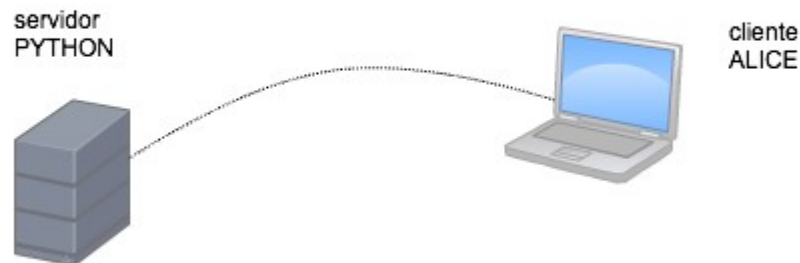
CAPITULO IX

SERVER SCRIPTING

CLIENTE SERVIDOR

Como definimos en nuestro marco teórico, debemos crear un servidor para que nuestra aplicación cliente (en este caso Alice) se comuniquen con nosotros.

Nuestro esquema de comunicación es el siguiente



Para este propósito debemos crear un socket para la comunicación entre procesos, en este caso es un servidor y lo vamos a asociar a nuestra dirección localhost (127.0.0.1), es decir, nuestra dirección local de la máquina donde correrá Alice, aunque esto es arbitrario y se podría ejecutar en cualquier servidor que tenga acceso a internet en el puerto que nosotros elijamos.

Del otro lado, del lado del cliente tambien se debera crear un socket cliente y asociarlo a la misma dirección y puerto.

Los sistemas operativos ya vienen con una serie de programas ejecutándose y por convención hay un rango de puertos que no se deben utilizar ya que están siendo utilizados. Este rango es el que va del 0 al 1023 por lo que tendremos puertos disponibles para elegir desde el 1024 a 65535. Elegiremos el puerto 10000 para nuestro servidor, nuestro esquema ahora tiene otro aspecto al ingresar la nueva especificación.

Creando un servidor

Llegó el momento de crear el servidor, esté dijimos que posee la siguiente información:

Direccion: 127.0.0.1

Puerto: 10000

Para esta tarea elegimos Python, el cual es un lenguaje de scripting que posee librerías de alto nivel con muchas funciones para facilitarnos la tarea, comenzamos importando un módulo, se realiza con la palabra reservada del lenguaje llamada import y luego procedemos a crear un socket de tipo:

AF_INET, nos indica la familia de protocolos que pueden comunicarse con nuestro servidor, el cual es ipv4.

SOCK_STREAM, setea el protocolo de transporte, esta familia crea un TCP, un protocolo orientado a conexión, con retransmisión de datos en caso de pérdida y aviso de lectura.

1. `import socket`
2. `# Creamos un TCP/IP socket`
3. `sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`
código de ./server.py

Ya tenemos una variable llamada `sock` lista para asociarla a una dirección y un puerto de la máquina para empezar a recibir conexiones.

4. `. server_address = ('localhost', 10000)`
5. `sock.bind(server_address)`

La asociacion antes mencionada la realizamos a traves de nuestro objeto `sock`, creado anteriormente invocando al metodo `.bind`, pasandole dos variables, la direccion la cual es `localhost` o `127.0.0.1` que es lo mismo y el puerto `10000`.

Ahora nos queda ingresar a un bucle infinito para recibir conexiones de clientes y realizarles la tarea que deseemos que realice nuestro servidor, el cliente nuestro cliente ALICE.

Ejemplo comunicación bidireccional. Alice - Servidor

Como ya hemos expuesto anteriormente tenemos un cliente listo para conectarse con un servidor, creímos conveniente realizar una comunicación bidireccional entre Alice y el servidor para que queden claros los conceptos antes expuestos.

El objetivo de este trabajo es una comunicación unidireccional y se verá más adelante este caso haremos una simple demostración.

Nuestro pequeño ejemplo consta de los siguientes pasos:

1. Nuestro servidor se levanta y espera conecciones.
2. Corremos una aplicación Alice que no hace más que enviarle información a nuestro servidor.
3. Alice muestra la información que recibe del servidor.
4. El servidor muestra la información que recibió de Alice.
5. FIN

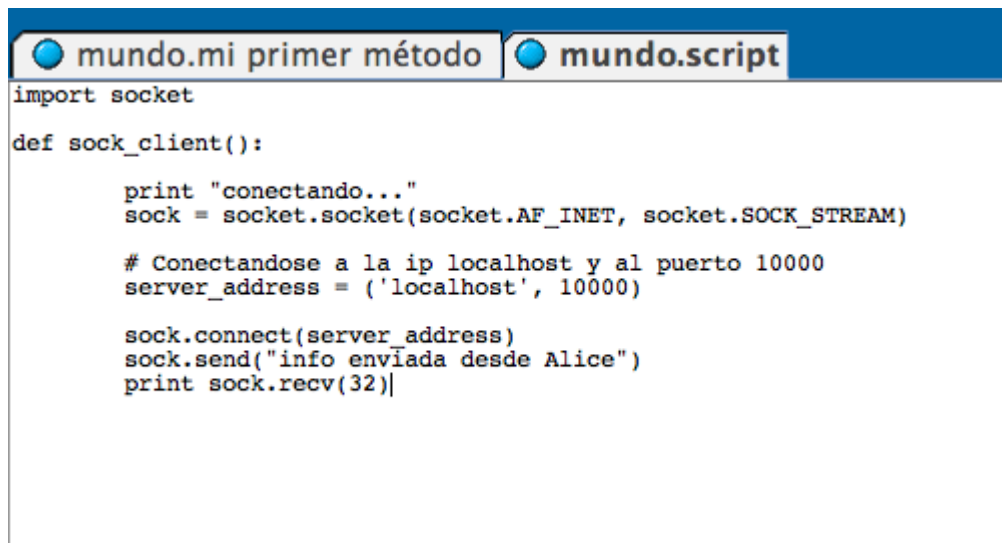
El servidor esperando conexiones va a lucir de la siguiente manera:

```
[(venv) ~/Documents/bkp/otros/tf/sock/ejemplo/ $ python server.py  
estoy corriendo en la siguiente ip, puerto  
( 'localhost', 10000)  
█
```

Y su código en el servidor es el siguiente:

```
1. import socket  
2.  
3. # Creación socket tcp/ip  
4. sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
5.  
6. # Asociando el socket a la ip y al puerto  
7. server_address = ('localhost', 10000)  
8. sock.bind(server_address)  
9. print ("estoy corriendo en la siguiente ip, puerto")  
10. print (server_address)  
11. sock.listen(1)  
12.  
13. while True:  
14.     connection, client_address = sock.accept()  
15.  
16.     try:  
17.         while True:  
18.             data = connection.recv(32)  
19.             print ("Recibi esta informacion...")  
20.             print (data)  
21.             connection.send(b'recibi tu info ALICE\n')  
22.         finally:  
23.             connection.close()
```

El cliente en Alice posee el siguiente código:



```
import socket

def sock_client():

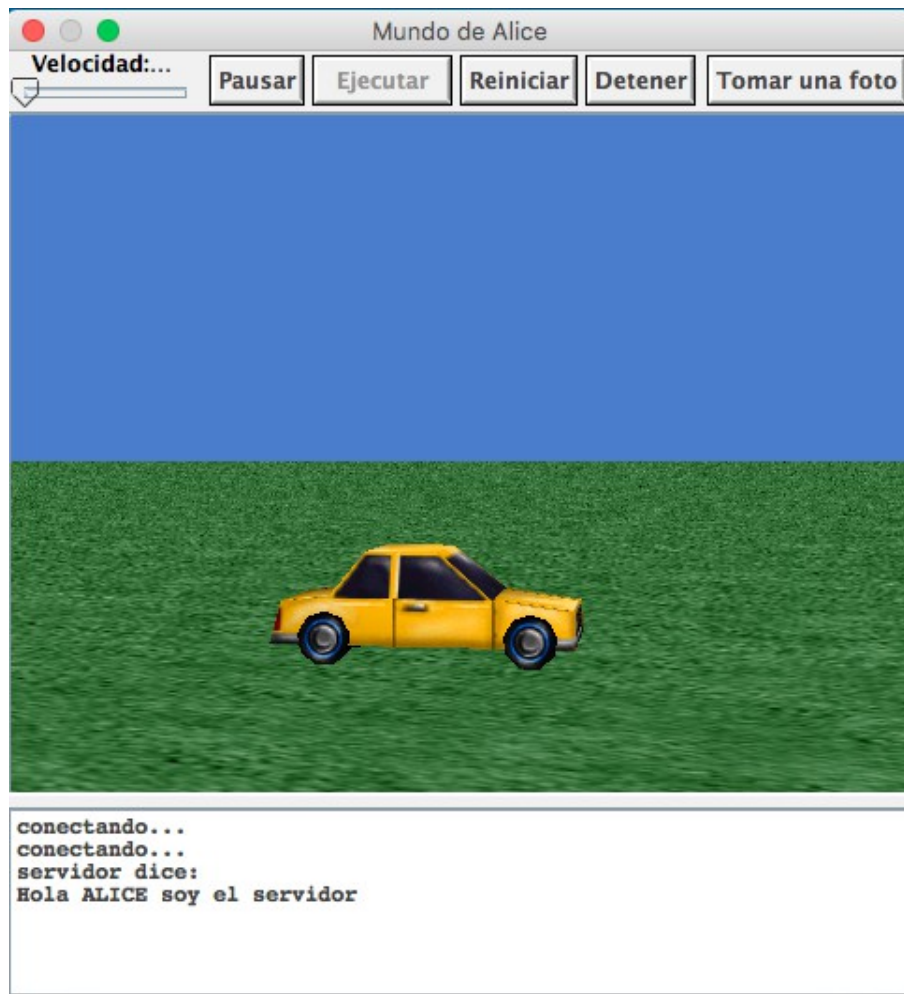
    print "conectando..."
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Conectandose a la ip localhost y al puerto 10000
    server_address = ('localhost', 10000)

    sock.connect(server_address)
    sock.send("info enviada desde Alice")
    print sock.recv(32)|
```

Solo resta queda correr Alice y ver sus resultados en ambas partes.

Desde Alice vemos que al ejecutar el proyecto obtenemos:



Y el servidor tambien recibio informacion:

```
(venv) ~/Documents/bkp/otros/tf/sock/ejemplo/ $ python server.py
estoy corriendo en la siguiente ip, puerto
('localhost', 10000)
Recibi esta informacion...
b'Hola desde Alice'
```

Esta fue una comunicaci3n bidireccional utilizando los m3todos **.send** para enviar y **recv** para recibir.

El servidor del proyecto Ana

Ahora hablaremos un poco acerca de la tarea que tiene encomendada el servidor del proyecto.

Es una pequeña funcionalidad la que realiza nuestro servidor y es por el cual fue construido. En el diagrama de bloque propuesto con el proyecto completo mencionamos una comunicación bluetooth entre el servidor y arduino, como vemos, nuestro servidor es el encargado de leer información desde Alice y reenviala a Arduino a través de Bluetooth.

En la máquina donde corre el servidor, tenemos puertos seriales por los que podemos comunicarnos. Uno de estos puertos seriales es el Bluetooth.

Los podemos listar en entornos Unix con el siguiente comando:

```
ls /dev/tty*
```

Esto nos muestra todos dispositivos seriales que podremos usar para establecer conexiones seriales, uno de estos es el llamado: **/dev/tty.HC-06-DevB**. Una vez obtenido dicho nombre, solo falta saber que la velocidad de comunicación de nuestro dispositivo Bluetooth conectado a Arduino es de **9600**, con estos parámetros podemos establecer una comunicación en nuestro servidor, con las siguientes líneas:

```
11. import serial
12.
13. #Serial port parameters
14. serial_speed = 9600
15. serial_port = '/dev/tty.HC-06-DevB' # bluetooth shield hc-06
16.
17. conn_bluetooth = serial.Serial(serial_port, serial_speed, timeout=1)
```

código de ./server.py

Como podremos observar ahora tenemos un objeto serial que está enlazado con nuestro Bluetooth y el Bluetooth de Arduino. Nos posibilita comunicarnos, es decir, enviar mensajes para que nuestro robot haga lo que le hemos indicado que haga. Esta variable en cuestión se llama **conn_bluetooth**.

Siguiendo con la configuración de nuestro servidor, nos queda setear un parámetro y es el cual le indicará al servidor, la cantidad de usuarios concurrentes que recibirá. Esto lo indicamos de la siguiente manera.

Al objeto **sock** anteriormente creado le invocaremos el metodo **.listen** con un entero, este número indicará la cantidad, ejemplo:

```
21. sock.listen(1)
```

código de ./server.py

Recibira una conexión concurrente. Ahora finalmente haremos que nuestro servidor ingrese en un ciclo de nunca terminar para que realice la tarea que le indicaremos.

En este ciclo realizaremos un enlace a nuestro cliente, es decir, cuando aparece un cliente lo aceptamos, si es que no hay ninguno asociado y luego al finalizar esta tarea lo excluimos de nuestra lista. Esto nos dará la posibilidad de hablar con distintos clientes pero siempre recordando que con uno a la vez.

```
22. while True:
```

```
23.     connection, client_address = sock.accept()
```

código de ./server.py

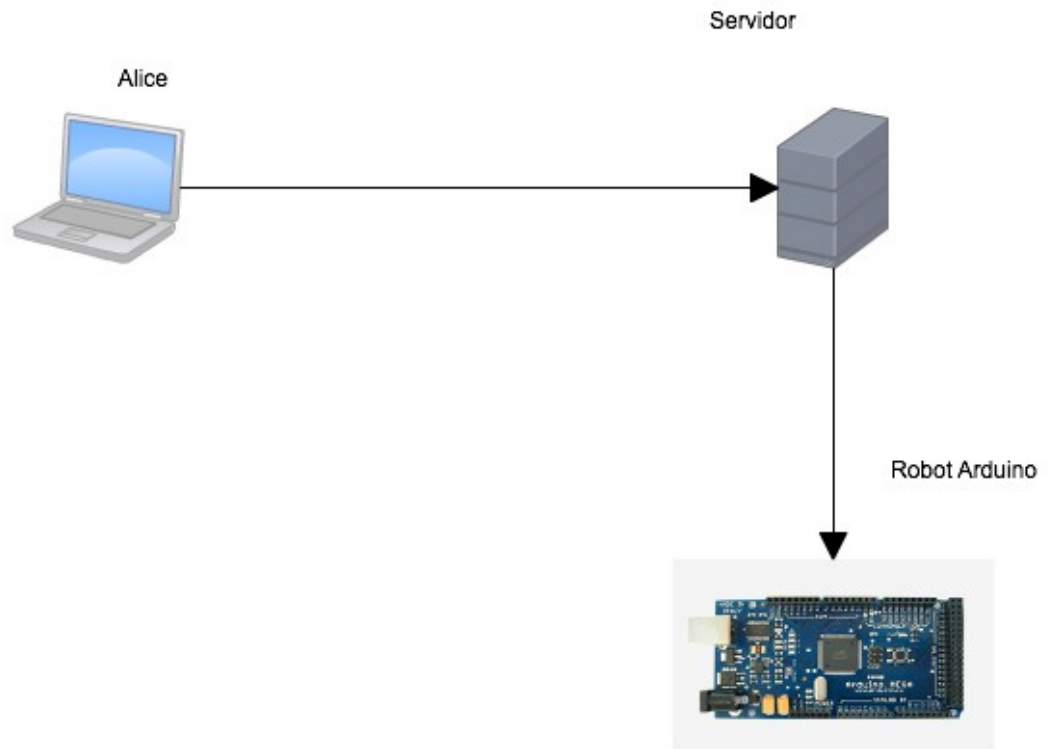
El objeto **connection** posee un método llamado **.recv**, el cual nos permitirá recibir información desde Alice.

Como podemos ver, tenemos dos descriptores para recibir o enviar información, a saber:

conn_bluetooth: Envía y recibe información al Bluetooth conectado a Arduino.

connection: Envía y recibe información a través del socket en localhost a Alice.

En nuestro servidor la comunicación fluye de la siguiente manera:



Por lo que los objetos antes mencionados solo utilizaran algunas de sus funcionalidades, a saber:

conn_bluetooth solo escribirá ya que el servidor le escribirá a Arduino, nunca en el otro sentido, como ejemplo, pondremos una escritura:

```
24. conn_bluetooth.write('1'.encode('ascii'))
```

código de ./server.py

En este caso enviamos en formato ascii el número 1 a nuestro robot-Arduino.

connection solo recibira informacion de Alice, ya que nuestro servidor en ningún momento le enviara informacion a alice, como ejemplo, pondremos una lectura:

```
25. connection.recv(16)
```

código de ./server.py

CAPITULO X

CREACIÓN DEL ROBOT

Nuestro robot por comodidad fue un auto. Sus movimientos van a ser muy sencillos ya que el motivo de este trabajo final no es del todo Robótica sino en brindar una interfaz entre Alice y un Robot.

Con el motivo facilitar la creación de un robot sencillo que se ejecute a través de Arduino elegimos realizar un auto que se mueva con motores de corriente continua.

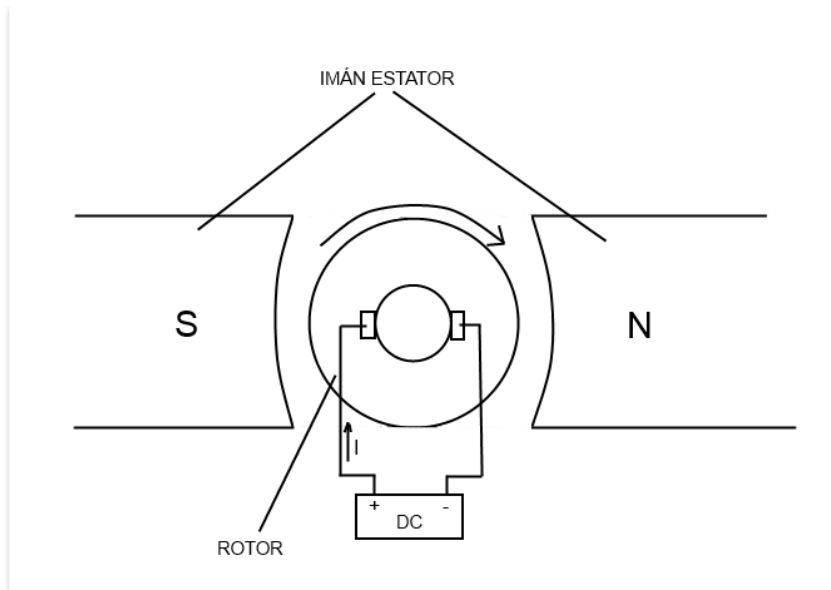
Un motor de corriente continua convierte la energía eléctrica en mecánica. Se compone de dos partes: el estator y el rotor.

El estator es la parte mecánica del motor donde están los polos del imán.

El rotor es la parte móvil del motor con devanado y un núcleo, al que llega la corriente a través de las escobillas.

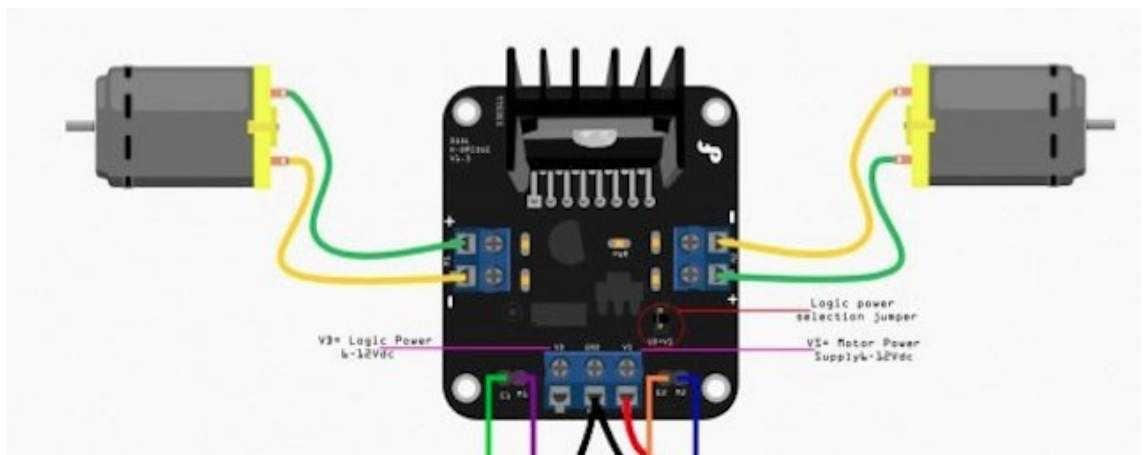
Cuando la corriente eléctrica circula por el devanado del rotor, se crea un campo electromagnético. Este interactúa con el campo magnético del imán del estator. Esto deriva en un rechazo entre los polos del imán del estator y del rotor creando un par de fuerza donde el rotor gira en un sentido de forma permanente.

Si queremos cambiar el sentido de giro del rotor, tenemos que cambiar el sentido de la corriente que le proporcionamos al rotor; basta con invertir la polaridad de la pila o batería.



d

Y como podemos observar fueron conectados a nuestro puente H de la siguiente manera:



El motor de la derecha simboliza las ruedas de la derecha, ya que ambas ruedas y por ende los motores están conectados en serie y lo mismo para el motor de la izquierda.

Cabe aclarar que la alimentación puede ser de distintas maneras pero nosotros hemos decidido conectar nuestros motores a 4 pilas AAA y nuestra placa Arduino a través del cable USB a un cargador de 5V y 1 A, como el de cualquier celular.

En el próximo capítulo cuando se muestre la interconexión lógica y física con Arduino explicaremos como se conecta el puente H y también como se alimenta la placa Arduino, ya que las pilas solamente fueron utilizadas para el puente que alimentaba a los motores, por una cuestión de rendimiento e independencia.

CAPITULO XI

ARDUINO

Introducción

En el capítulo anterior vimos como se creo nuestro robot pero omitimos las conexiones a Arduino porque creímos pertinente hacerlo en este capítulo que es donde también vamos a programar la placa para que realice los movimientos.

En este capítulo veremos todo lo relacionado a Arduino.

En una primera instancia vamos a proponer descargar el software necesario para Arduino y su correcta instalación, luego explicaremos la conexión física de nuestro robot con Arduino y por último crearemos el código necesario en el ide de Arduino para que funcionen nuestros motores y el Bluetooth y lo cargaremos a la placa.

Descarga Arduino

La descarga es por sistema operativo, debemos dirigirnos a la siguiente sitio web

<https://www.arduino.cc/en/Main/Software>

Y seleccionamos la versión adecuada a nuestro sistema operativo, en este caso lo haremos para Mac OS pero también están sus versiones para Microsoft Windows y Linux.

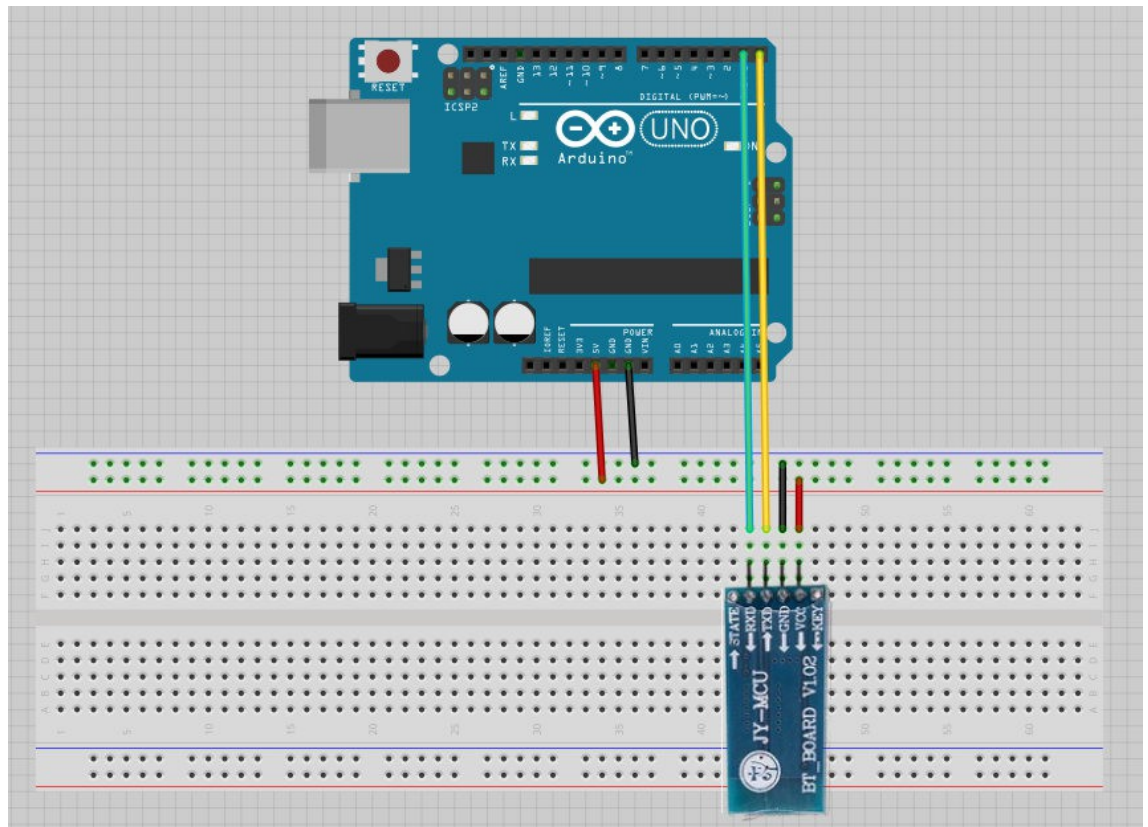
En Mac OS debemos descomprimir el zip y luego arrastrar el icono a Aplicaciones y listo, ya queda instalada la aplicación.

En WIndows debemos descomprimir y luego ejecutarlo cada vez que lo vayamos a utilizar, el archivo se llama “arduino-1.0.5-windows.exe”.

Si todo fue bien veremos el siguiente IDE:

Conexiones Robot y Arduino

Primero vamos a configurar nuestro Bluetooth con la placa Arduino. Poseemos el bluetooth HC 06 con 4 pines, para lograr la configuración nos basaremos en los pines TX y RX de nuestra placa Arduino, hay que tener en cuenta con esos pines (TX y RX) ya que son los mismos que utiliza el cable usb para cargarle datos a Arduino, por este movito cuando vayamos a cargar el software a Arduino debemos desconectar el Bluetooth.



La interconexion de los pines del Bluetooth con Arduino queda:

Bluetooth HC 06	Arduino
PIN GND	PIN GND
PIN VCC	3.5 V
PIN TX	PIN RX
PIN RX	PIN TX

La conexion de los motores es en el puente H L298

Como anteriormente mencionamos, los motores de cada lado vienen en serie, es decir, las ruedas del lado derecho del auto tienen cada una un motor y estos motores están en serie, las del otro lado tambien.

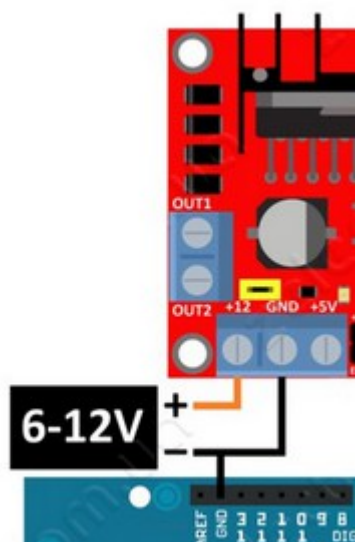
Dicho esto nos debería quedar una conexión de la siguiente manera:

Motores Derecha	Motores Izquierda
INI1	INI3
INI2	INI4

Además el puente H se enlaza a Arduino del siguiente modo:

Puente H L298	ARDUINO
INI1	PIN 2
INI2	PIN 3
INI3	PIN 5
INI3	PIN 4

Solo nos queda conectar nuestro puente H a la Alimentación que la realizamos a través de un cofre con 4 pilas AAA con la siguiente conexión:



PUENTE H L298	Cofre Pilas / Arduino
PIN +12V	positivo cofre pilas
PIN GND	negativo cofre pilas / PIN GND ARDUINO

Programando Arduino

La programación en Arduino es bastante sencilla, primero explicaremos como leemos a través del Bluetooth.

Como lo conectamos a los pin TX y RX de nuestro Arduino, no debemos realizar nada dentro de nuestro código Arduino ya que estaremos usando el puerto serial por defecto, solamente debemos indicarle a Arduino la velocidad de transferencia en baudios de nuestro dispositivo Bluetooth.

Para programar Arduino tenemos que tener en cuenta lo siguiente,

Un código para Arduino posee 2 métodos indispensables y una declaración de constantes.

Estos metodos se llaman **setup()** y **loop()**, explicaremos cual es su objetivo.

En el método **Setup()**, es donde se indica que entradas lógicas y digitales vamos a utilizar, entonces, siguiendo con lo antes expuesto, nuestro Bluetooth debería ser declarado ahí.

```
arduino
int IN1 = 2;
int IN2 = 3;
int IN3 = 5;
int IN4 = 4;

void setup()
{
  Serial.begin(9600);
  pinMode (IN4, OUTPUT);
  pinMode (IN3, OUTPUT);
  pinMode (IN2, OUTPUT);
  pinMode (IN1, OUTPUT);
}
```

Serial.begin(9600); está haciendo referencia a nuestro Bluetooth.

Vemos también en nuestro código que existen variables llamadas IN1, IN2, etc. Estas son las declaradas anteriormente para nuestro puente H.

Su manera de ser vistas por Arduino es declarandolas de esa forma, a saber:

```
pinMode(IN1, OUTPUT);
```

Como podemos apreciar, nuestro método setup es bastante sencillo.

Ahora veremos como esta formado nuestro método **loop()**.

En este método se realiza la tarea, es un bucle como su nombre lo indica y por ende se ejecuta indefinidamente, se debe tener cuidado de no caer en bucles sin sentido.

Generalmente este método está chequeando los datos que aparecen en nuestra variable Serial o algun otro factor que cambie del exterior y en base a ese estado, realiza una acción.

arduino §

```
void loop(){

    while (Serial.available() > 0) {

        char inChar = (char)Serial.read();
        switch(inChar) {
            case '0':
                digitalWrite (IN4, LOW);
                digitalWrite (IN1, LOW);
                digitalWrite (IN3, LOW);
                digitalWrite (IN2, LOW);
                Serial.print("Se detiene");
                break;
            case '1':
                digitalWrite (IN4, LOW);
                digitalWrite (IN1, LOW);
                digitalWrite (IN3, HIGH);
                digitalWrite (IN2, HIGH);
                Serial.print("Marcha Atras...");
                break;
            case '2':
                digitalWrite (IN4, HIGH);
                digitalWrite (IN1, HIGH);
                digitalWrite (IN3, LOW);
                digitalWrite (IN2, LOW);
                Serial.print("Marcha adelante");
                break;
        }
    }
}
```

Como podemos ver, existe una lectura constante a nuestra variable Serial y en base a ese valor realizamos una acción.

Al poner alguno de nuestros motores en Low o en High, nuestros motores se mueven en un sentido u otro.

Eso es todo lo que hace nuestro robot, sus 3 movimientos que son recibidos desde el servidor.

Vemos que si el servidor no da información, el robot se queda ejecutando en el último estado que recibió.

CAPITULO XII

EJEMPLO IMPLEMENTACIÓN EN CONJUNTO

A continuación explicaremos acerca de nuestra implementación, en el informe hemos cubierto varios aspectos acerca de cómo llevamos a cabo nuestro proyecto por lo que esto sería un resumen y mostrar como nos quedó nuestra aplicación programada en Alice.

Al levantar nuestro servidor y esperar conexiones de Alice, este realiza una conexión a Arduino a través de Bluetooth para que en cuanto le llegue información desde Alice se la pueda trasladar a Alice.

Por lo que nuestro servidor ahora deberá escribir en el descriptor del Bluetooth, la conexión y la escritura a expondremos a continuación:

```
# # puerto serial parametros
serial_speed = 9600
serial_port = '/dev/tty.HC-06-DevB' # bluetooth shield hc-06
#connect to bluetooth
conn_bluetooth = serial.Serial(serial_port, serial_speed, timeout=1)

conn_bluetooth.write('1'.encode('ascii'))
```

Los movimientos aceptados por el servidor son los siguientes:

Alice	Servidor	Arduino
Comienza y envía un 5	recibe un 5, no envía nada	no recibe nada
Se detiene y envía un 0	recibe un 0 y envía un 0	recibe un 0
Marcha atrás y envió un 1	recibe un 1 y envía un 1	recibe un 1
Avanza y envía un 2	recibe un 2 y envía un 2	recibe un 2

Observamos en el ejemplo que estamos enviando un '1', por lo que nuestro robot debería moverse hacia adelante.

Cubierto esta funcionalidad, resta mirar como funciona Alice para enviar esta comunicación al servidor y complementariamos la aplicación.

Cuando el mundo comienza, Alice realiza la conexión al servidor a través del método único de Scripting llamado **move_arduino()** y como vimos le envía una 5 al servidor, el cual este no realiza ningún movimiento, solamente se enlaza para que el servidor quede como puente entre Alice y Arduino.

En la imagen podrán apreciar de que manera Alice realiza esta acción.

Cuando el mundo se está ejecutando, hacer durante:

world.move

Eventos

Cuando el Mundo comienza, hacer **world.my first method**

Cuando el Mundo se está ejecutando

Comienzo: **<Ninguno>**

Durante: **world.move**

Teminar: **<Ninguno>**

Mientras **↑** es apretado

Comienzo: **<Ninguno>**

Durante: **world.forward**

Teminar: **world.stop-forward**

Mientras **↓** es apretado

Comienzo: **<Ninguno>**

Durante: **world.reverse**

Teminar: **world.stop-reverse**

Y **world.move** posee la siguiente instrucción:

Guión (Script) **move_arduino("5")**

Luego vemos que tenemos dos métodos más, el de la flecha hacia arriba y la flecha hacia abajo.

Vamos a explicar uno, ya que ambos realizan lo mismo pero en distinto sentido.

La instrucción dice Mientras el cursor sea *hacia arriba* es presionado y tenemos 3 opciones:

a- Comienzo: Este no realiza ninguna acción.

b- Durante: invoca a **world.forward**

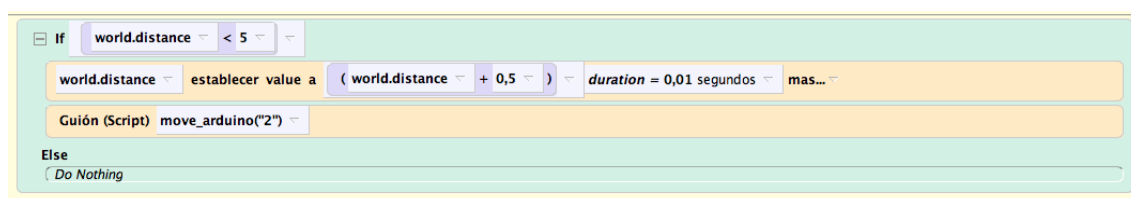
c- Terminar: invoca a **world.stop-reverse**

Explicaremos uno por uno estos métodos, a saber:

world.forward como su nombre en inglés lo sugiere, realiza el movimiento hacia adelante.

Lo realiza en Alice y enviando información a nuestro servidor, aquí es donde sucede la magia del tiempo real.

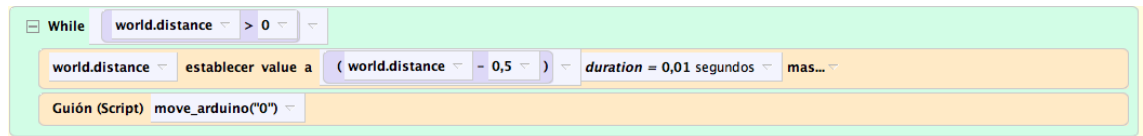
Examinaremos este método:



Como pueden observar válida la condición si la variable **world.distance** es menor a 5 y en caso de ser verdadera le suma a world.distance el valor de 0,5.

Esto lo realiza para que el auto vaya incrementando su posición en la aplicación Alice, es netamente de Alice pero en la línea siguiente vemos que invoca al método **move_arduino** con el valor '2' y como dijimos, este método hace que el robot avance hacia adelante.

El otro método que nos estaría faltando explicar es **world.stop-reverse**, como ya se pueden imaginar, expondremos este método de Alice y lo analizaremos.



El condicional aca es otro, es mientras **world.distance** sea mayor a cero, restale a **world.distance** 0.5, esto es para que sea gradual el frenado dentro de la aplicación Alice.

La siguiente línea le escribe a nuestro servidor un cero para que detenga al robot.

Con esto completamos la aplicacion, ya que Arduino fue programado en la sección anterior con los movimientos que va a recibir desde el servidor.

Para finalizar, les mostramos una pantalla de nuestro escenario con el auto, listo para avanzar hacia adelante o hacia atras a traves de las teclas de navegación.



RESULTADOS

Los resultados obtenidos son alentadores ya que hubo momentos donde se veía imposible conectar Alice con otro software del exterior. Queda en el aprendizaje del alumno la enseñanza de que con investigación se pueden lograr grandes cosas, ya que al momento de la propuesta se veía como un proyecto muy difícil de concretar.

En el transcurso del desarrollo fuimos intercambiando información con colegas y todos se veían entusiasmados por lo que este proyecto podría alcanzar.

Se le hicieron demostraciones a alumnos de secundarios y las preguntas que ellos realizaban eran bastante motivadoras ya que se noto el interes que este proyecto generaba.

Con el fin de lograr conquistar el público del secundario es una herramienta bastante llamativa ya que baja conceptos que parecen muy abstractos si no se tiene conocimiento.

Como contra debemos tratar de realizar un proyecto similar con menores costos.

Los costos en el día de hoy cuando estamos escribiendo este informe de resultados (abril del año 2016) son los siguientes:

Material	Precio
Arduino Mega	\$400 (pesos Argentinos)
Bluetooth HC 06	\$155 (pesos Argentinos)
Puente H L298	\$120 (pesos Argentinos)
Protoboard	\$70 (pesos Argentinos)
Cables	\$50 (pesos Argentinos)
Pilas recargables con cargador	\$450 (pesos Argentinos)
Cofre pilas	\$50 (pesos Argentinos)

Motor con rueda (se utilizaron 4)	\$150 (\$600) (pesos Argentinos)
TOTAL	\$1895

Estos precios fueron consultados en www.mercadolibre.com.ar el día 16/04/2016, por lo que dependiendo de la fecha y el lugar, habría que agregarle el envío y la diferencia inflacionaria, además el alumno debería poseer una notebook.

Creemos en un proyecto más económico por lo que realizamos otro presupuesto con menos cosas pero que cumple la misma funcionalidad, por si alguien está interesado en crear algo similar al proyecto Ana.

Material	Precio
Kit Arduino UNO+cables+protoboard	\$440 (pesos Argentinos)
Motores con rueda X 2	\$300 (pesos Argentinos)
Rueda unica multiple movimiento	\$50 (pesos Argentinos)
TOTAL	\$790

Vemos que al suprimir algunas funcionalidades podemos bajar los costos, cabe aclarar que este presupuesto también fue hecho en Mercadolibre y que la comunicación y alimentación van a ser obtenidas por el cable usb, eliminando el Bluetooth y las pilas, lo cual le brinda más libertad.

No investigamos en cómo alimentar mediante paneles solares o energías alternativas, queda para quien quiera lograr bajar los costos de este trabajo, como así también realizar todo con un solo motor.

CONCLUSIONES

Las conclusiones que tenemos son que es posible la realización de proyectos con fines educativos. Las consecuencias de no conseguir incentivar a los que están por comenzar carreras universitarias son grandes y en base a proyectos como Ana demuestran que es posible salirse de la educación típica.

Con apoyo de docentes es posible realizar talleres en el último año de la secundaria y lograr explayarse más en el tema.

Muchas de las personas con las que hablamos creen que la informática es arreglar computadoras, algo que además de ser aburrido para todo el mundo, es algo que se puede hacer sin estudiar.

Nuestra misión es mostrar que la programación es divertida y no es rutinaria.

BIBLIOGRAFÍA

Documentación

Trabajo final de Mauricio, Cáceres Universidad de Mendoza, “Desarrollo e implementación de un brazo robótico manejado por la aplicación Alice”

Documentación digital

“Alice”, <http://www.alice.org/index.php>.

“Programación dedicada a objetos”.

http://es.wikipedia.org/wiki/Programación_orientada_a_objetos.

“Robótica”, apuntes de catedra Ing. Celeste D’Inca.

“Robótica”, http://es.wikipedia.org/wiki/Brazo_robótico.

“Robótica Educativa”, http://www.roboticaeducativa.com/rob_edu.php.

http://es.wikipedia.org/wiki/Robótica_educativa.

“Cultura y educación en la sociedad de la información”.

<https://books.google.com.ar/books?id=ZB9KsyL0vaAC&lpg=PA275&dq=ntics&hl=es&pg=PA275#v=onepage&q=ntics&f=false>.

“Bluetooth HC 06”, <http://ingerick.weebly.com/arduino/bluetooth-hc-06-configuracin-con-arduino>.

“Python - Arduino”, <http://www.toptechboy.com/arduino/python-with-arduino-lesson-2-installing-the-software/>

“Alfabetización siglo XXI”, <http://www.pagina12.com.ar/diario/sociedad/3-216461-2013-03-24.html>

“Puente H y motores DC”, <http://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/>