UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

UNIDAD DE GESTIÓN DE TECNOLOGÍAS

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN.

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

TRABAJO DE TITULACIÓN PARA LA OBTENCIÓN DEL TÍTULO DE:

TECNÓLOGO EN ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

TEMA: "IMPLEMENTACIÓN DE DOS ROBOTS SUMO MEDIANTE ARDUINO PARA PRÁCTICAS DE ROBÓTICA EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS".

AUTOR: CBOP. SUQUILLO LEON CHRISTIAN DAVID

DIRECTOR: Ing. Pablo Pilatasig

LATACUNGA

2017

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN Y AVIÓNICA

CERTIFICACIÓN

Certifico que el trabajo de Titulación,**"IMPLEMENTACIÓN DE DOS ROBOTS SUMO MEDIANTE ARDUINO PARA PRÁCTICAS DE ROBÓTICA EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS"**realizado el Sr. **CBOP. SUQUILLO LEON CHRISTIAN DAVID**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor **CBOP. SUQUILLO LEON CHRISTIAN DAVID** para que lo sustente públicamente.

Latacunga, 22 de Febrero del 2017

Atentamente,

Ing. Pablo Pilatasig

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN Y AVIÓNICA

AUTORÍA DE RESPONSABILIDAD

Yo, **CBOP. SUQUILLO LEON CHRISTIAN DAVID,** con cédula de identidad Nº 1002812145, declaro que este trabajo de titulación **"IMPLEMENTACIÓN DE DOS ROBOTS SUMO MEDIANTE ARDUINO PARA PRÁCTICAS DE ROBÓTICA EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 22 de Febrero 2017.

_____

Suquillo León Christian David

C.I: 1002812145

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN Y AVIÓNICA

AUTORIZACIÓN (PUBLICACIÓN BIBLIOTECA VIRTUAL)

Yo, **CBOP. SUQUILLO LEON CHRISTIAN DAVID,** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"IMPLEMENTACIÓN DE DOS ROBOTS SUMO MEDIANTE ARDUINO PARA PRÁCTICAS DE ROBÓTICA EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS** "cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 22 de Febrero 2017.

_____

Suquillo León Christian David

C.I: 1002812145

DEDICATORIA

Me gustaría ofrecer este logro a toda mi familia.

Para mis padres María y Guillermo, por su ayuda y comprensión en todos los momentos de mi vida que siempre están pendientes de mí y que me han enseñado que solo para la muerte no hay solución.

Me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, todo ello con una gran dosis de amor, a mi hermano Vinicio, que siempre me ha apoyado en el transcurso de mi vida.

Doy gracias a nuestro padre celestial por darme una hermosa familia, mi esposa Nataly y a mis hijos.

Hijos, son mi motivación, impulsándome a vencer cualquier adversidad que se presentan en mi camino, y me impulsan día tras día a cumplir nuestras metas propuestas.

AGRADECIMIENTO

Este logro con lleva una gran gratitud hacia Dios, quien con su bendición me ha permitido llegar a alcanzar una meta más en el transcurso académico.

Para mis padres María y Guillermo, por su ayuda y comprensión en todos los momentos de mi vida que siempre están pendientes de mí y que me han enseñado que solo para la muerte no hay solución.

Me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, y todo ello con una gran dosis de infinito amor, a mi hermano Vinicio, que siempre me ha apoyado en el transcurso de mi vida.

Doy gracias a nuestro padre celestial por darme una hermosa familia, mi esposa Nataly y a mis hijos.

Hijos, son mi motivación, impulsándome a vencer cualquier adversidad que se presentan en mi camino, y me impulsan día tras día a cumplir las metas propuestas.

Con amor y agradecimiento infinito.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El desarrollo del este proyecto técnico consiste en la implementación de dos robots Zumo 32U4 marca Pololu, con dos micro motoreductores de metal de 75:1 HP y el otro con la relación de 50:1 HP. Cada robot necesita de 4 pilas AA, en este proyecto se utilizó las pilas marca Maxday de 4700 mAh de níquel hidruro metálico (Ni-Mh). Este robot es compatible con Arduino Leonardo por lo que para su programación en el IDE de Arduino se debe seleccionar esta tarjeta. Se comprobó el funcionamiento de los tres leds de usuario, presentes en la placa principal del robot Zumo, mediante un programa que haga parpadear los tres leds al mismo tiempo. Se realizó un programa para visualizar texto en las dos líneas del LCD incluido en la placa principal del robot Zumo, este LCD es de 8x2. Existen tres botones de usuario llamados A, B y C, mediante programación se ejecutó la tarea que cuando se pulse el botón A, se encienda el led Amarillo, cuando se pulse el botón B, se encienda el led verde y cuando se pulse el botón C se encienda el led rojo. Como integración de los elementos del robot se crearon dos programas, el primero que realiza el giro del robot de derecha e izquierda hacia adelante y hacia atrás, el otro para realizar la detección y seguimiento de objetos con la ayuda del sensor de proximidad.

Palabras Claves

ROBOT ZUMO
MICRO MOTOREDUCTORES
ARDUINO
SENSOR DE PROXIMIDAD
LEDS Y BOTONES

ABSTRACT

The development of this technical project consists in the implementation of two robots PolumoZumo 32U4, with two micro metal engine reductors of 75: 1 HP and the other with the ratio of 50: 1 HP. Each robot needs 4 AA batteries; in this project we used the Maxday brand batteries of 4700 mAh nickel metal hydride (Ni-Mh). This robot is compatible with Arduino Leonardo so for programming in the Arduino IDE you must select this card. The operation of the three user leds, present in the main plate of the robot Zumo, was verified by means of a program that blinks the three leds at the same time. A program was performed to visualize text in the two lines of the LCD included in the main plate of the robot Zumo, this LCD is of 8x2. There are three user buttons named A, B and C, programmatically executed the task that when the A button is pressed, the yellow led lights up, when the B button is pressed, the green led lights up and when the button C is pressed the red LED lights up. Also, the buzzer, proximity sensor and operation of the two motors were also checked. As integration of the elements of the robot, two programs were created, the first to make the robot turn right and left forward and backward, the other to perform the detection and tracking of objects with the help of the proximity sensor.

Keywords

ROBOT ZUMO
MICRO MOTOREDUCTORS
ARDUINO
PROXIMITY SENSOR
LEDS AND BUTTONS

CHECKED BY:
Lcda. MARÍA ELISA COQUE
ENGLISH TEACHER UGT.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA.

IMPLEMENTACIÓN DE DOS ROBOTS SUMO MEDIANTE ARDUINO PARA PRÁCTICAS DE ROBÓTICA EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS.

1.1 ANTECEDENTES.

En la provincia de Cotopaxi, cantón de Latacunga se encuentra ubicada la Unidad de Gestión de Tecnologías de la Universidad de las Fuerzas Armadas (ESPE), mencionada institución está orientada a la enseñanza der la educación superior en el nivel tecnológico superior, apoyados en material técnico de sus múltiples laboratorios promoviendo el aprendizaje, de manera teórica- practica el manejo y utilización de dispositivos que van acorde a la vanguardia de la tecnología.

Es Así que se han realizado trabajos relacionados a robótica como el de (Llanos &Lliguin, 2010) plantea el diseño e implementación del sistema que está enfocado en un robot zoomórfico de tipo cuadrúpedo, similar a un perro, que posee una plataforma de aluminio dispuesta de articulaciones formadas por 15 servomotores, por consiguiente posee 15 grados de libertad; la aplicación utiliza el módulo procesador de voz denominado VRbot para receptar las órdenes emitidas por el administrador

Otro trabajo presentado por la Universidad Tecnológica de Mixteca, su autor (González, 2014) con el Tema: “Diseño y Construcción de un Robot Sumo”), su objetivo era reducir el costo del robot, con referencia a otros ya existentes en el mercado. Aunque la mayor parte del diseño fue determinada por las características físicas del chasis empleado (estructura mecánica), las ideas de la parte electrónica como usos de puentes H, reguladores de voltaje, ATMEGA8 (la parte de control, uso de ADC’s, implementación de rutinas) pueden ser aplicadas a cualquier robot sumo con ligeras

modificaciones. Este diseño es sencillo y funcional, y con esto se cumplen los objetivos generales y específicos planteados para este proyecto.

El desarrollo del presente trabajo es importante porque dotará al laboratorio de Instrumentación Virtual y al club de robótica el material didáctico que permitirá tener acceso al manejo y control de robots.

1.2 PLANTEAMIENTO DEL PROBLEMA.

La robótica avanza de forma exponencial, lo que exige a las instituciones de educación superior una actualización continua para formar profesionales con competencias acorde a la demanda laboral y así poder desenvolverse de forma óptima.

Mediante trabajos de graduación se están incorporando nuevos dispositivos a los laboratorios como material didáctico para ayudar en el proceso de enseñanza aprendizaje; actualmente no existen robots sumo mediante arduino en la institución, incumpliendo con los requerimientos necesarios para adquirir conocimiento en robótica y su correcto funcionamiento, originándose dificultades en la investigación y desarrollo de nuevos prototipos en base a ya existentes.

Es preciso que el laboratorio de Instrumentación Virtual de la Unidad de Gestión de Tecnologías y el club de robótica, cuente con dispositivos necesarios para desarrollar prácticas de laboratorio con los estudiantes de la carrera de Electrónica mención Instrumentación y Aviónica y el club de robótica.

1.3 JUSTIFICACIÓN

La implementación de este trabajo de graduación dotará a la carrera de Electrónica mención Instrumentación y Aviónica de robots con tecnología actual, de manera que se puedan realizar prácticas de laboratorio en lo referente al manejo y control de la robótica.

Lo que permitirá a los estudiantes, docentes e integrantes del club de robótica desarrollar habilidades y destrezas en la manipulación de robots tipo Sumo, teniendo así un prototipo base y poder diseñar e implementar otros

robots con mejores prestaciones, incentivando de esta manera a la investigación.

El proyecto es factible porque los equipos necesarios para su implementación se pueden encontrar en el mercado nacional, y sus respectivos manuales de operación.

1.4 OBJETIVOS.

1.4.1 Objetivo general.

Implementar dos robots sumos mediante arduino para prácticas de robótica en el Laboratorio de Instrumentación Virtual de la Unidad de Gestión de Tecnologías de la Universidad de las Fuerzas Armadas - ESPE.

1.4.2 Objetivos específicos.

- Indagar las características de los robots sumo controlados por arduino a través de la investigación bibliográfica documental.
- Configurar el robot sumo mediante arduino para la lectura y escritura de datos.
- Realizar pruebas de funcionamiento del proyecto finalizado.

1.5 Alcance.

El presente proyecto técnico consiste en implementar dos robots sumo mediante arduino para prácticas de robótica, lo cual permitirá a los estudiantes y docentes de la carrera de Electrónica mención Instrumentación y Aviónica e integrantes del Club de robótica, ampliar las habilidades y destrezas en el manejo y control de este tipo de dispositivos.

CAPÍTULO II

MARCO TEÓRICO.

2.1 Robótica

Según Romero Costas 2012:

> La robótica es la ciencia que estudia el diseño y la implementación de robots, conjugando múltiples disciplinas, como la mecánica, la electrónica, la informática, la inteligencia artificial y la ingeniería de control, entre otras.

2.1.1 Robot

Según la Asociación Japonesa de Robótica Industrial (**JIRA**)**:**

> Los robots son dispositivos capaces de moverse de modo flexible, análogo al que poseen los organismos vivos, con o sin funciones intelectuales, lo que permite la realización de operaciones en respuesta a órdenes recibidas por humanos.(Zabala, 2007)

Según el Instituto de Robótica de Norteamérica (**RIA**):

> Define a un robot industrial como un manipulador multifuncional y reprogramable diseñado para desplazar materiales, componentes, herramientas o dispositivos especializados por medio de movimientos programados variables, con el fin de realizar diversas tareas.(Zabala, 2007)

2.1.2 Tipos de Robots

Algunas de las clasificaciones son:

Según el uso del robot

- Industriales
- Espaciales
- Médicos
- Domésticos
- Sociales
- Agrícolas

Según el medio en que se desarrolla la actividad

- Acuáticos
- Terrestres
- Aéreos
- Híbridos

Según la ubicación de la inteligencia del robot

Zabala 2007 describe como:

- **Autónomos:** la inteligencia está ubicada en el mismo robot. Puede comunicarse con otros o con un sistema central, pero los aspectos esenciales de funcionamiento se resuelven en forma independiente en el propio robot.
- **Control automatizado** (semiautónomos): la mayor parte de la inteligencia del robot está ubicada en un sistema central. Los sensores pueden ser locales, es decir que le envían la información obtenida a ese sistema central, o globales. El sistema central les comunica a los robots las acciones que deben realizar. Un ejemplo de este modelo es la categoría Microsoft de fútbol de robots de la FIRA.
- **Híbridos:** son robots autónomos que, en ciertos momentos del proceso, pueden ser controlados por humanos o por un sistema central. Un ejemplo son los robots que se utilizan en misiones espaciales, que operan en forma autónoma pero que, ante un percance, pueden ser dirigidos desde nuestro planeta. (Zabala, 2007)

2.1.3 Unidades de un Robot

Según Zabala 2007:

> En la arquitectura de cualquier computadora, se encuentra las siguientes unidades que la componen:
>
> - **Unidades de procesamiento:** es el conjunto de dispositivos que se encargan de realizar la transformación de los datos de entrada para obtener los datos de salida.
> - **Unidades de entrada:** son las unidades que permiten realizar el ingreso de información para su posterior procesamiento.

- **Unidades de salida:** son las unidades que se ocupan de comunicarle los resultados del procesamiento al usuario u operador.

Un robot tiene la misma arquitectura. Las unidades de entrada de un robot son los sensores, que pueden ser externos, como un sensor de tacto, o internos, como un encoder que permite determinar la distancia recorrida por una rueda. A las unidades de salida se las conoce como actuadores, como son leds de señalización, buzzers, motores, displays, etc. En síntesis, el robot recibe información del ambiente mediante sus sensores, procesa la información con su unidad de procesamiento y realiza sus acciones al mover motores y encender luces y buzzers. Por ejemplo uno de los robots más conocidos es Terminator. Cuyos sensores son las cámaras que le permiten mirar, su sistema auditivo y los sensores de tacto que tiene su piel. Sus actuadores esenciales son los motores o los músculos de alambre que conforman su cuerpo.

Uno de los problemas más apasionantes de la robótica es el equilibrio que es necesario obtener entre las tres unidades, para lograr que el robot cumpla con su objetivo. Por ejemplo, los sensores más sofisticados o que entregan mayor cantidad de datos, como puede ser una cámara de video, exigen de parte del procesador un mayor tiempo de trabajo para poder obtener un conjunto de información que resulte significativo. De la misma manera, el control de los actuadores sofisticados, como cierto tipo de motores, consume tiempo de procesamiento que es absolutamente necesario para que el robot opere en tiempo real. En síntesis, es imprescindible lograr el equilibrio entre velocidad y precisión, en especial en aquellos robots que operan en entornos muy dinámicos. Es por eso que, habitualmente, se utilizan ciertos procesadores específicos para el filtrado de la información de entrada y para el control de los actuadores, y así se libera de esta tarea al procesador central y se complementa su función. (Zabala, 2007)

2.2 Robótica Educativa

Según López 2013:

> La robótica educativa se integra a paso acelerado en las instituciones educativas a nivel mundial y en nuestro país; situación ante la cual no pueden estar ajenos los docentes. Esta disciplina que aborda el diseño, implementación y programación de robots, se integra como una herramienta multidisciplinar que, además de trabajar sobre contenidos curriculares de materias como Ciencias, Matemáticas, Física o Tecnología, favorece la formación de otras competencias esenciales para el progreso académico de los estudiantes, acordes a los postulados de la Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura (UNESCO) saber ser saber hacer saber convivir y aprender a transformarse uno mismo y la sociedad.
>
> La robótica educativa se basa en los principios pedagógicos del constructivismo, el construccionismo, el enfoque histórico cultural y el desarrollo de aprendizajes significativos. El construccionismo se fundamenta en el uso de las TICs en la educación. Otorga a los estudiantes un rol activo como diseñadores y constructores de sus propios proyectos y su aprendizaje, aprendizaje que se apropia del entorno, lo imagina, lo simula, lo crea, lo recrea y lo innova, proyectando al estudiante, como plantea Vygotsky, a la zona de desarrollo próximo. Esto permite un salto cualitativo en la educación, el traspaso de la instrucción tradicional a la formación con sentido, es decir, un aprehender el mundo social, cultural, científico y tecnológico.(López, 2013)

2.3 Robot Zumo 32U4

El robot Zumo 32U4 (Figura 1) es un robot completo y versátil controlado por un Microcontrolador ATmega32U4 compatible con Arduino. Cuando se ensambla el robot de seguimiento mide menos de 10 cm en cada lado, haciéndolo adecuado para competiciones de Mini-Sumo (Corporation, 2015).

Figura 1 Robot Zumo 32U4

Fuente: (Corporation, 2015)

Según Corpotation 2015:

> El corazón del Zumo 32U4 se encuentra un micro controlador AVR ATmega32U4 integrado de Atmel (Figura 2), junto con dos controladores de puente H que accionan los motores del robot. El robot también cuenta con una variedad de sensores, incluyendo codificadores de cuadratura y sensores inerciales (acelerómetro y giróscopo) en la placa principal, junto con reflectores y sensores de proximidad en el conjunto de sensores delanteros. Los botones incorporados ofrecen una interfaz conveniente para la entrada del usuario, y un LCD, un zumbador, y los indicadores LED permiten que el robot proporcione retroalimentación.(Corporation, 2015)

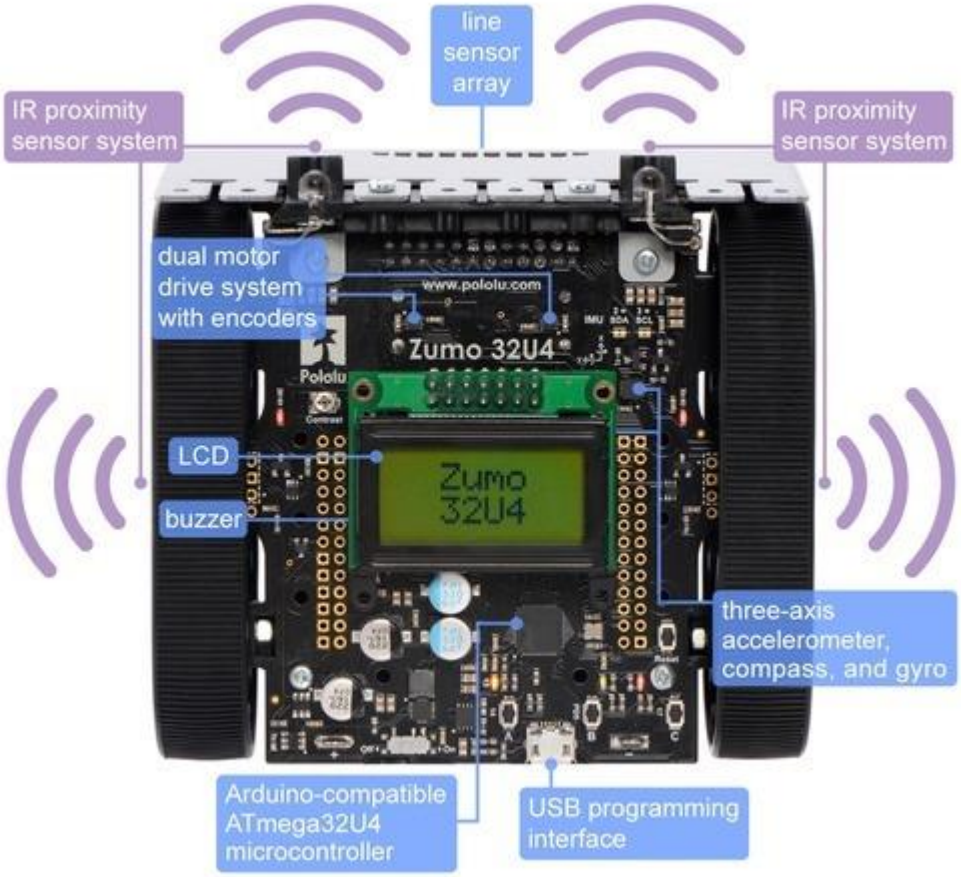

Figura 2 Principales características del robot Zumo 32U4

Fuente: (Corporation, 2015)

2.3.1 Microcontrolador

La placa principal 32U4 del robot Zumo incorpora un microcontrolador AVR ATmega32U4 de Atmel, USB integrado y sincronizado por un oscilador de cristal de precisión de 16 MHz.

La placa principal incluye un conector USB Micro-B que se puede utilizar para conectar al puerto USB de una computadora a través de un cable USB A a Micro-B. La conexión USB se puede utilizar para transmitir y recibir datos desde el ordenador y programar la tarjeta a través de USB (Corporation, 2015).

El ATmega32U4 de Zumo viene precargado con el mismo gestor de arranque USB compatible con Arduino, lo que le permite ser programado fácilmente con el IDE de Arduino(Corporation, 2015).

2.3.2 Interface de Usuario

El robot Zumo 32U4 tiene 8 leds indicadores.

Según Corporation 2015:

> Un LED amarillo de usuario que se conecta al pin digital 13 del Arduino o PC7.El Bootloader del robot Zumo atenúa y enciende este LED mientras espera que se cargue un programa.
>
> Un LED de usuario verde se conecta a PD5 y se enciende cuando el pin esta en nivel bajo. Mientras la placa está ejecutando el Bootloader o un programa está siendo compilado en el entorno Arduino, destellará este LED cuando esté transmitiendo datos a través de la conexión USB.
>
> Un LED de usuario rojo está conectado al pin 17 del Arduino, o PB0, y se enciende cuando el pin esta en nivel bajo. Mientras la placa está ejecutando el Bootloader o un programa está siendo compilado en el entorno Arduino, destellará este LED cuando esté transmitiendo datos a través de la conexión USB.
>
> Dos LEDs rojos en los bordes izquierdo y derecho de la placa indican cuando emisores de infrarrojos del robot son activos en el lado correspondiente.
>
> Dos LEDs de potencia azules bajo las esquinas traseras de la placa principal indican cuando el robot está recibiendo alimentación de las baterías (el interruptor de alimentación debe estar encendido). El LED izquierdo está conectado al voltaje de la batería (VBAT), mientras que el LED derecho está conectado a la salida del regulador de 5 V de la placa.
>
> Un LED verde bajo el borde posterior centro de la placa principal indica cuando la tensión del bus USB (VBUS) está presente.(Corporation, 2015)

Pulsadores

Según Corporation 2015:

> El Zumo 32U4 tiene cuatro botones: un botón de reinicio en el borde derecho y tres pulsadores de usuario que se encuentra A lo largo del

borde trasero de la placa principal. Los pulsadores del usuario, marcados A, B y C, están en los pines del Arduino 14 (PB3), PD5 y pin 17 (PB0), respectivamente. Al pulsar uno de estos botones, el pin de entrada/salida asociado se conecta a tierra a través de una Resistencia.(Corporation, 2015)

También posee un LCD de 8x2 caracteres y un zumbador que sirve para generar sonidos simples y música. De forma predeterminada, está conectado al pin digital 6.

2.3.3 Motores

Existen dos controladores de motores de Texas Instruments DRV8838 (figura 3) incorporados que alimentan a los micros motoreductores de metal (figura 4).

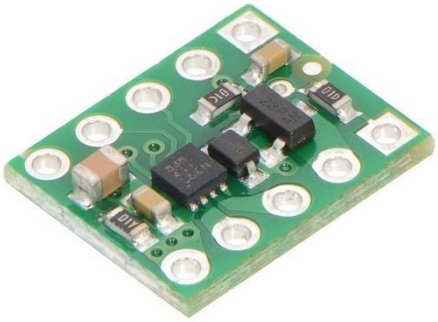

Figura 3 Controlador de motores

Fuente: (Corporation, 2015)

Figura 4 Micro motoreductores de metal

Fuente: (Corporation, 2015)

Según Corporation 2015:

Se utilizan cuatro pines del Arduino para los controladores:

- **pin digital 15,** o PB1, controla la dirección del motor derecho.
- **pin digital 16,** o PB2, controla la dirección del motor izquierdo**.**
- **pin digital 9,** o PB5, controla la velocidad del motor derecha con PWM (modulación por ancho de pulso)
- **pin digital 10,** o PB6, controla la velocidad del motor izquierdo con PWM

2.3.4 Codificadores de cuadratura

Cada motor de accionamiento del Zumo 32U4 tiene un sistema de codificación en cuadratura (Figura ***5***) correspondiente de un disco magnético unido al motor de eje extendido y un par de sensores de efecto Hall montados debajo de la placa principal(Corporation, 2015).

Los codificadores proporcionan una resolución de 12 cuentas por vuelta del motor en ambos canales. Para calcular las cuentas por revolución de las ruedas motrices, multiplicar relación de transmisión cajas de cambio por 12. Por ejemplo de un motor de 75:1 que de acuerdo a la hoja de datos en forma precisa es 75.81:1 se debe multiplicar 75.81x12 que da como resultado de 909,72 cuantas por revolución (CPR).

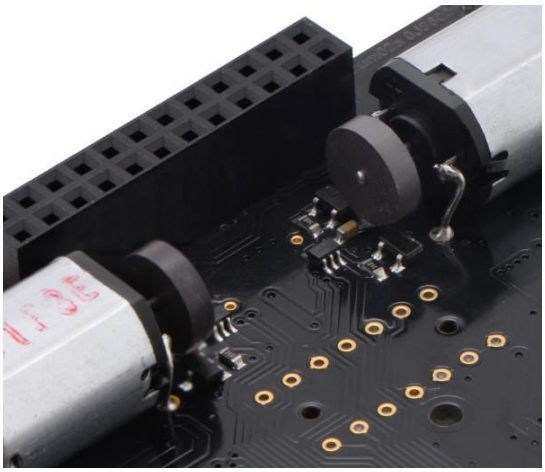

Figura 5 Codificadores en cuadratura

Fuente: (Corporation, 2015)

2.3.5 Conjunto de sensores frontales de línea y proximidad

Según Corporation 2015:

> El Zumo 32U4 Front Sensor Array es una placa que se conecta a la placa principal. La placa cuenta con cinco sensores de línea y tres sensores de proximidad, aunque por defecto, sólo puede tener seis estos ocho sensores conectados al Microcontrolador en un momento dado.
>
> Los cinco sensores de línea ayudan al Zumo a distinguir superficies entre luz y oscuridad. También pueden utilizarse para detectar fuentes de luz infrarroja, como el sol. Cada sensor consiste de un LED de emisor infrarrojo orientado hacia abajo (IR) junto con un fototransistor que puede detectar el reflejo la luz infrarroja del LED.
>
> Los tres sensores de proximidad pueden ayudar a detectar objetos cercanos. También se pueden utilizar para detectar comandos de los controles remotos IR típicos. (Corporation, 2015)

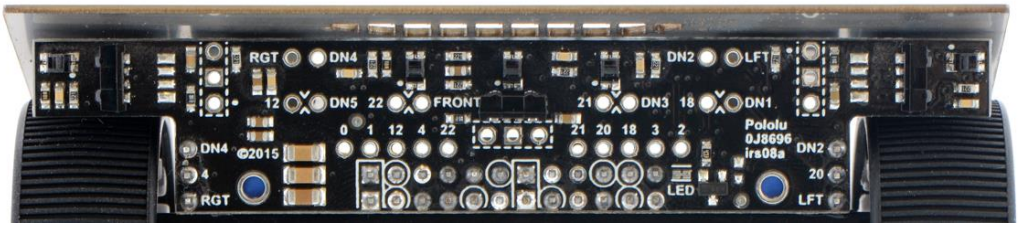

Figura 6 Array de sensores frontales

Fuente: (Corporation, 2015)

2.3.6 Detección de proximidad

La placa principal tiene cuatro emisores IR (Figura 7) :

- Los LEDs IR izquierda y derecha están montados en superficie a ambos lados del Zumo entre las ruedas. Emiten luz a la izquierda y a la derecha.
- Los LED IR delanteros izquierdo y derecho están orientados hacia el frente. Estos LEDs están incluidos, pero que deben ser instalados por el usuario.

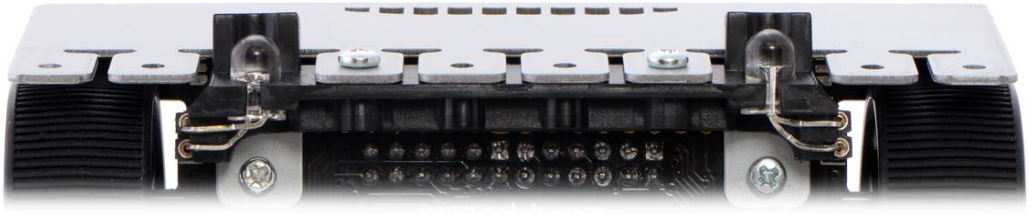

Figura 7 Sensores IR frontales

Fuente: (Corporation, 2015)

2.3.7 Sensores inerciales

Según Corporation 2015:

> El Zumo 32U4 incluye sensores inerciales a bordo que pueden utilizarse en aplicaciones avanzadas, como detecta colisiones y determina su propia orientación mediante la implementación de una Unidad Medición Inercial (IMU).
>
> El primer chip (figura 8), un módulo de brújula ST LSM303D, combina un Acelerómetro de 3 ejes y magnetómetro de 3 ejes en un solo paquete. El segundo chip (figura 9) es un ST L3GD20H Giroscopio de 3 ejes. Ambos chips de sensores comparten un bus I2C conectado al Interfaz I2C de ATmega32U4.(Corporation, 2015)

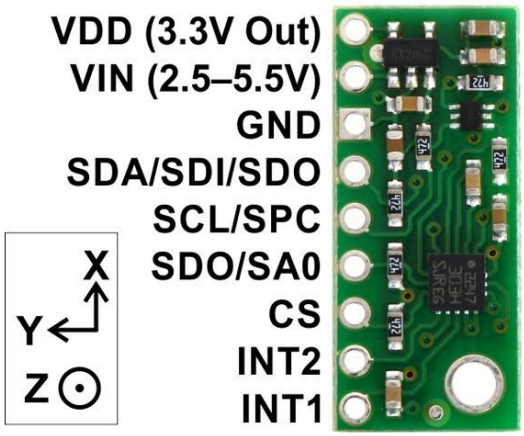

Figura 8 Acelerómetro y compas

Fuente: (Corporation, 2015)

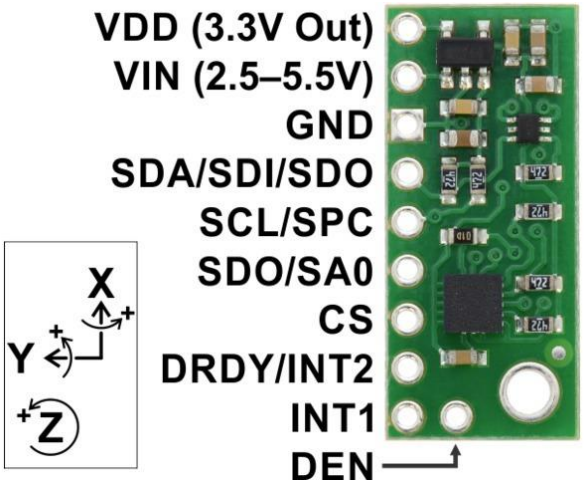

Figura 9 Giróscopo de 3 ejes

Fuente: (Corporation, 2015)

2.3.8 Alimentación

El chasis Zumo tiene un compartimento interno para cuatro pilas AA. Se recomienda el uso de pilas AA recargables Ni MH (Figura 10), lo que da como resultado una tensión nominal de 4,8 V (1,2 V por célula). También se puede utilizar pilas alcalinas, que nominalmente le daría 6 V (Corporation, 2015).

Figura 10 Pilas AA recomendadas

Fuente: (Corporation, 2015)

Según Corporation 2015:

> El terminal negativo de la batería está conectado a GND. El terminal positivo de la batería se denomina VB. VB alimenta a un circuito que proporciona protección inversa y conmutación de alimentación

controlada por el interruptor de alimentación de la placa. Los Salida de este circuito se denomina VBAT.

VBAT suministra energía a los motores a través de los controladores de motor DRV8838, por lo que los motores sólo funcionaran si las pilas están instaladas y el interruptor de encendido está en la posición "On".

El voltaje de la batería en VBAT puede ser monitoreado a través de un divisor de voltaje que está conectado al pin analógico 1 (PF6) de forma predeterminada. (Corporation, 2015)

Regulador de voltaje de 5V

Según Corporation 2015:

VBAT suministra energía a un regulador de 5V basado en el Convertidor de Texas Instruments TPS63061. El regulador funciona con una tensión de entrada de 2.7 V a 11.8 V y tiene una eficacia típica de 80% a 90% para la mayoría de las combinaciones de tensión de entrada y carga.

Figura 11 Regulador de voltaje de 5V

Fuente: (Corporation, 2015)

Selección de alimentación

Según Corporation 2015:

El circuito de selección de potencia de la placa principal Zumo 32U4 utiliza el multiplexor de potencia TPS2113A de Texas Instruments para elegir si su suministro de lógica de 5 V (designado 5V) se obtiene de USB o las baterías a través del regulador, lo que permite al robot realizar la transición de forma segura y sin problemas.

El TPS2113A está configurado para seleccionar la alimentación de batería regulada (VREG) a menos que la salida del regulador caiga por debajo de unos 4,5 V. Si esto sucede, seleccionará la más alta de las dos fuentes, que típicamente será la tensión de bus USB 5 V si el Zumo está Conectado a USB. (Corporation, 2015)

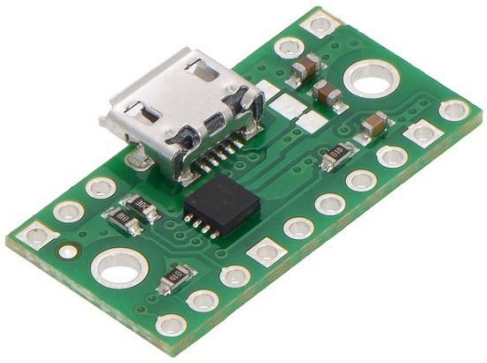

Figura 12 Multiplexor de fuentes de alimentación

Fuente: (Corporation, 2015)

El Zumo 32U4 está diseñado para ser programado por USB desde el IDE de Arduino.

2.4 Arduino

2.4.1 Definición

Arduino es en realidad tres cosas.

Según Torrente 2013:

> Una placa hardware libre que incorpora un microcontrolador reprogramable y una serie de pines-hembra (los cuales están unidos internamente a las patillas de E/S del microcontrolador) que permiten conectar allí de forma muy sencilla y cómoda diferentes sensores y actuadores.
>
> La “placa hardware” es una PCB (del inglés “printed circuit board”, o sea, placa de circuito impreso). Las PCBs son superficies fabricadas de un material no conductor (normalmente resinas de fibra de vidrio reforzada, cerámica o plástico) sobre las cuales aparecen laminadas

(“pegadas”) pistas de material conductor (normalmente cobre). Las PCBs se utilizan para conectar eléctricamente, a través de los caminos conductores, diferentes componentes electrónicos soldados a ella. Una PCB es la forma más compacta y estable de construir un circuito electrónico (en contraposición a una breadboard, perfboard o similar) pero, al contrario que estas, una vez fabricada, su diseño es bastante difícil de modificar. Así pues, la placa Arduino no es más que una PCB que implementa un determinado diseño de circuitería interna.

Se debe especificar el modelo concreto de “placa Arduino”, ya que existen varios modelos oficiales, cada una con diferentes características (como el tamaño físico, el número de pines-hembra ofrecidos, el modelo de microcontrolador incorporado –y como consecuencia, entre otras cosas, la cantidad de memoria utilizable–, etc.). Conviene conocer estas características para identificar qué placa Arduino es la que nos convendrá más en cada proyecto.

Los microcontroladores incorporados en las diferentes placas Arduino pertenecen todos a la misma “familia tecnológica”, por lo que su funcionamiento en realidad es bastante parecido entre sí. En concreto, todos los microcontroladores son de tipo AVR, una arquitectura de microcontroladores desarrollada y fabricada por la marca Atmel. Es por eso que, en este documento se nombrao “placa Arduino” a cualquiera de ellas mientras no sea imprescindible hacer algún tipo de distinción.

El diseño hardware de la placa Arduino está inspirado originalmente en el de otra placa de hardware libre preexistente, la placa Wiring Esta placa surgió en 2003 como proyecto personal de Hernando Barragán, estudiante por aquel entonces del Instituto de Diseño de Ivrea (lugar donde surgió en 2005precisamente la placa Arduino). (Torrente, 2013)

Según Torrente 2013:

Un software (más en concreto, un “entorno de desarrollo”) gratis, libre y multiplataforma (ya que funciona en Linux, MacOS y Windows) que se debe instalar en el ordenador con el fin de escribir, verificar y guardar (“cargar”) en la memoria del microcontrolador de la placa Arduino el conjunto de instrucciones a ejecutar. Es decir programarlo, la manera estándar de conectar nuestro computador con la placa, Arduino para poder enviar y grabar dichas instrucciones es mediante un simple cable USB, gracias a que la mayoría de placas Arduino incorporan un conector de este tipo.

Los proyectos Arduino pueden ser autónomos o no. En el primer caso, una vez programado su microcontrolador, la placa no necesita estar conectada a ningún computador y puede funcionar autónomamente si dispone de alguna fuente de alimentación. En el segundo caso, la placa debe estar conectada de alguna forma permanente (por cable USB, por cable de red Ethernet, etc.) a un computador ejecutando algún software específico que permita la comunicación entre este y la placa y el intercambio de datos entre ambos dispositivos. Este software específico se debe programar generalmente mediante algún lenguaje de programación estándar como Python, C, Java, Php, etc., y será independiente completamente del entorno de desarrollo Arduino, el cual no se necesitará más, una vez que la placa ya haya sido programada y esté en funcionamiento.(Torrente, 2013)

Según Torrente 2013:

Un lenguaje de programación libre. Por “lenguaje de programación” se entiende cualquier idioma artificial diseñado para expresar instrucciones (siguiendo unas determinadas reglas sintácticas) que pueden ser llevadas a cabo por máquinas. Concretamente dentro del lenguaje Arduino, se encuentran elementos parecidos a muchos otros lenguajes de programación existentes (como los bloques condicionales, los bloques repetitivos, las variables, etc.), así como también diferentes comandos–asimismo llamados “órdenes” o “funciones” – que permiten especificar de una forma coherente y sin

errores las instrucciones exactas programadas en el microcontrolador de la placa. Estos comandos se escriben mediante el entorno de desarrollo Arduino.

Tanto el entorno de desarrollo como el lenguaje de programación Arduino están inspirado en otro entorno y lenguaje libre preexistente: Processing, desarrollado inicialmente por Ben Fry y Casey Reas.

Que el software Arduino se parezca tanto a Processing no es casualidad, ya que este está especializado en facilitar la generación de imágenes en tiempo real, de animaciones y de interacciones visuales, por lo que muchos profesores del Instituto de Diseño de Ivrea lo utilizaban en sus clases. Como fue en ese centro donde precisamente se inventó Arduino es natural que ambos entornos y lenguajes guarden bastante similitud. No obstante, hay que aclarar que el lenguaje Processing está construido internamente con código escrito en lenguaje Java, mientras que el lenguaje Arduino se basa internamente en código C/C++.

Con Arduino se pueden realizar multitud de proyectos de rango muy variado: desde robótica hasta domótica, pasando por monitorización de sensores ambientales, sistemas de navegación, telemática, etc. Realmente, las posibilidades de esta plataforma para el desarrollo de productos electrónicos son prácticamente infinitas y tan solo están limitadas por nuestra imaginación.(Torrente, 2013)

2.4.2 Origen de Arduino

Según Torrente 2013:

Arduino nació en el año 2005 en el Instituto de Diseño Interactivo de Ivrea(Italia), centro académico donde los estudiantes se dedicaban a experimentar con la interacción entre humanos y diferentes dispositivos (muchos de ellos basados en microcontroladores) para conseguir generar espacios únicos, especialmente artísticos. Arduino apareció por la necesidad de contar con un dispositivo para utilizaren las aulas que fuera de bajo coste, que funcionase bajo cualquier

sistema operativo y que contase con documentación adaptada a gente que quisiera empezar de cero.

La idea original fue, pues, fabricar la placa para uso interno de la escuela.

No obstante, el Instituto se vio obligado a cerrar sus puertas precisamente en

2005. Ante la perspectiva de perder en el olvido todo el desarrollo del proyecto

Arduino que se había ido llevando a cabo durante aquel tiempo, se decidió liberarlo y abrirlo a "la comunidad" para que todo el mundo tuviera la posibilidad de participaren la evolución del proyecto, proponer mejoras y sugerencias y mantenerlo "vivo". Y así ha sido: la colaboración de muchísima gente ha hecho que Arduino poco a poco haya llegado a ser lo que es actualmente: un proyecto de hardware y software libre de ámbito mundial.

El principal responsable de la idea y diseño de Arduino, y la cabeza visible del proyecto es el llamado "ArduinoTeam", formado por MassimoBanzi (profesor en aquella época del Instituto Ivrea), David Cuartielles (profesor de la Escuela de Artes y Comunicación de la Universidad de Mälmo, Suecia), David Mellis (por aquel entonces estudiante en Ivrea y actualmente miembro del grupo de investigación High-LowTech del MIT Media Lab), Tom Igoe (profesor de la Escuela de Arte Tisch de NuevaYork), y Gianluca Martino.

2.4.3 Arduino como software libre

Torrente 2013 menciona que:

Según la Free Software Foundation (http://www.fsf.org), organización encargada de fomentar el uso y desarrollo del software libre a nivel mundial, un software para ser considerado libre ha de ofrecer a cualquier persona u organización cuatro libertades básicas e imprescindibles:

- **Libertad 0:** la libertad de usar el programa con cualquier propósito y en cualquier sistema informático.

- **Libertad 1:** la libertad de estudiar cómo funciona internamente el programa,
- y adaptarlo a las necesidades particulares. El acceso al código fuente es un requisito previo para esto.
- **Libertad 2:** la libertad de distribuir copias.
- **Libertad 3:** la libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Un programa es software libre si los usuarios tienen todas estas libertades. Así pues, el software libre es aquel software que da a los usuarios la libertad de poder ejecutarlo, copiarlo y distribuirlo (a cualquiera y a cualquier lugar), estudiarlo, cambiarlo y mejorarlo, sin tener que pedir ni pagar permisos al desarrollador original ni a ninguna otra entidad específica. La distribución de las copias puede ser con o sin modificaciones propias, y atención, puede ser gratis ¡o no!: el "software libre" es un asunto de libertad, no de precio.

Para que un programa sea considerado libre a efectos legales ha de someterse a algún tipo de licencia de distribución, entre las cuales se encuentran la licencia GPL (General PublicLicense), o la LGPL, entre otras. El tema de las diferentes licencias es un poco complicado: hay muchas y con muchas cláusulas. Para saber más sobre este tema, se puede consultar http://www.opensource.org/licenses/category,donde está disponible el texto oficial original de las licencias más importantes.

Ejemplos de software libre hay muchos: el kernel Linux, el navegador Firefox, la suite ofimática Libre Office, el reproductor multimedia VLC, etc.

El software Arduino es software libre porque se publica con una combinación de la licencia GPL (para el entorno visual de programación propiamente dicho) y la licencia LGPL (para los códigos fuente de gestión y control del microcontrolador a nivel más interno).

La consecuencia de esto es, en pocas palabras, que cualquier persona que quiera (y sepa), puede formar parte del desarrollo del software Arduino y contribuir así a mejorar dicho software, aportando nuevas características, sugiriendo ideas de nuevas funcionalidades, compartiendo soluciones a posibles errores existentes, etc. Esta manera de funcionar provoca la creación espontánea de una comunidad de personas que colaboran mutuamente a través de Internet, y consigue que el software Arduino evolucione según lo que la propia comunidad decida. Esto va mucho más allá de la simple cuestión de si el software Arduino es gratis o no, porque el usuario deja de ser un sujeto pasivo para pasar a ser (si quiere)un sujeto activo y partícipe del proyecto.(Torrente, 2013)

2.4.4 Arduino como hardware libre

Según Torrente 2013:

El hardware libre (también llamado “open-source” o “de fuente abierta”) comparte muchos de los principios y metodologías del software libre. En particular, el hardware libre permite que la gente pueda estudiarlo para entender su funcionamiento, modificarlo, reutilizarlo, mejorarlo y compartir dichos cambios. Para conseguir esto, la comunidad ha de poder tener acceso a los ficheros esquemáticos del diseño del hardware en cuestión (que son ficheros de tipo CAD). Estos ficheros detallan toda la información necesaria para que cualquier persona con los materiales, herramientas y conocimientos adecuados pueda reconstruir dicho hardware por su cuenta sin problemas, ya que consultando estos ficheros se puede conocer qué componentes individuales integran el hardware y qué interconexiones existen entre cada uno de ellos.

La placa Arduino es hardware libre porque sus ficheros esquemáticos están disponibles para descargar de la página web del proyecto con la licencia Creative Commons Attribution Share-Alike, la cuales una licencia libre que permite realizar trabajos derivados tanto personales como comerciales (siempre que estos den crédito a Arduino y

publiquen sus diseños bajo la misma licencia). Así pues, uno mismo se puede construir su propia placa Arduino “a mano”. No obstante, lo más normal es comprarlas de un distribuidor ya pre ensamblados y listos la placa Arduino, aunque sea libre, no puede ser gratuita, ya que es un objeto físico y su fabricación cuesta dinero.

A diferencia del mundo del software libre, donde el ecosistema de licencias libres es muy rico y variado, en el ámbito del hardware todavía no existen prácticamente licencias específicamente de hardware libre, ya que el concepto de “hardware libre” es relativamente nuevo. De hecho, hasta hace poco no existía un consenso generalizado en su definición. Para empezar a remediar esta situación, en el año 2010 surgió el proyecto OSHD (http://freedomdefined.org/OSHW), el cual pretende establecer una colección de principios que ayuden a identificar como “hardware libre” un producto físico. OSHD no es una licencia (es decir, un contrato legal), sino una declaración de intenciones (es decir, una lista general de normas y de características) aplicable a cualquier artefacto físico para que pueda ser considerado libre. El objetivo de la OSHD (en cuya redacción ha participado gente relacionada con

el proyecto Arduino, entre otros) es ofrecer un marco de referencia donde se respete por un lado la libertad de los creadores para controlar su propia tecnología y al mismo tiempo se establezcan los mecanismos adecuados para compartir el conocimiento y fomentar el comercio a través del intercambio abierto de diseños. En otras palabras: mostrar que puede existir una alternativa a las patentes de hardware que tampoco sea necesariamente el dominio público. El proyecto OSHD abre, pues, un camino que crea precedentes legales para facilitar el siguiente paso lógico del proceso: la creación de licencias libres de hardware.

El objetivo del hardware libre es, por lo tanto, facilitar y acercar la electrónica, la robótica y en definitiva la tecnología actual a la gente, no de una manera pasiva, meramente consumista, sino de manera

activa, involucrando al usuario final para que entienda y obtenga más valor de la tecnología actual e incluso ofreciéndole la posibilidad de participar en la creación de futuras tecnologías.

Básicamente, el hardware abierto significa tener la posibilidad de mirar qué es lo que hay dentro de las cosas, y que eso sea éticamente correcto. Permite, en definitiva, mejorar la educación de las personas. Por eso el concepto de software y hardware libre es tan importante, no solo para el mundo de la informática y de la electrónica, sino para la vida en general.(Torrente, 2013)

2.4.5 IDE de Arduino

Dado que el Arduino es un pequeño ordenador que ejecuta una serie de códigos que previamente se han cargado a la propia placa mediante un programa. Este programa se llama IDE (Figura 13**)**, que significa "Integrated Development Environment" ("Entorno de Desarrollo Integrado"). Este IDE estará instalado en nuestro PC, es un entorno muy sencillo de usar y en él escribiremos el programa que queramos que el Arduino ejecute. Una vez escrito, lo cargaremos a través del USB y Arduino comenzará a trabajar de forma autónoma.

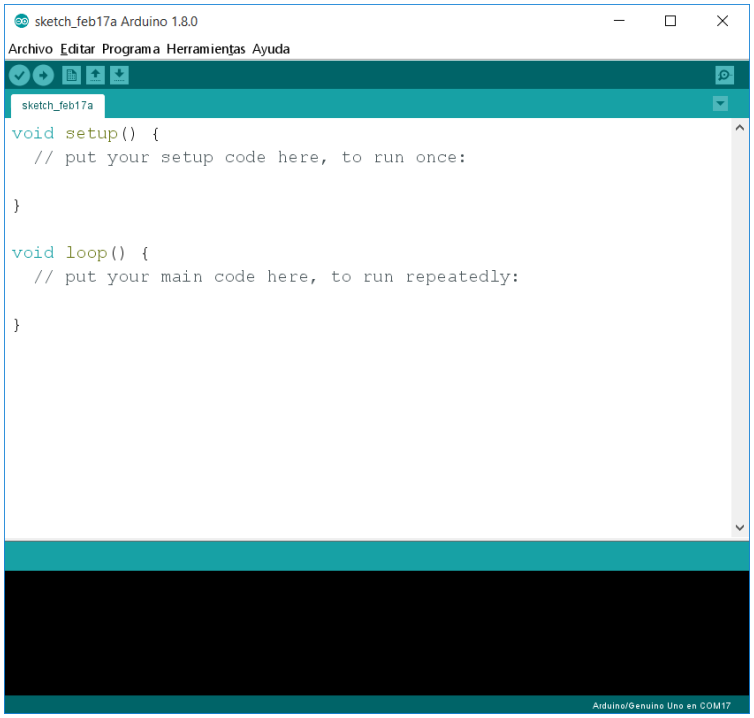

Figura 13 IDE de Arduino

2.5 Librería Zumo32U4

Esta es una biblioteca de C ++ para el IDE de Arduino que ayuda a acceder al hardware a bordo del robot Zumo 32U4.

El robot Zumo 32U4 consiste en el Zumo chasis, la placa principal 32U4 Zumo, y la matriz de sensores de frente Zumo 32U4. Cuenta con un microcontrolador integrado AVR ATmega32U4, controladores de motor, codificadores, sensores de proximidad, sensores de línea, zumbador, cuatro botones, conectores LCD, acelerómetro LSM303D, y el giroscopio L3GD20H.

2.5.1 Instalación de la librería

Descargar el archivo de notas más reciente de GitHub y descomprimirlo.

- Cambiar el nombre de la carpeta "zumo-32u4-Arduino-libreria-master" a "Zumo32U4".
- Mover la carpeta "Zumo32U4" en el directorio "librerías" dentro de su directorio Arduino. Puede ver su ubicación, abra el menú "Archivo" y seleccionar "Preferencias" en el IDE de Arduino. Si no hay ya una carpeta "librerias" en ese lugar.
- Después de instalar la biblioteca, reiniciar el IDE de Arduino.

2.5.2 Clases y funciones

Las principales clases y funciones proporcionadas por la librería se enumeran a continuación:

- L3G
- LSM303
- Zumo32U4ButtonA
- Zumo32U4ButtonB
- Zumo32U4ButtonC
- Zumo32U4Buzzer
- Zumo32U4Encoders
- Zumo32U4IRPulses

- Zumo32U4LCD
- Zumo32U4LineSensors
- Zumo32U4Motors
- Zumo32U4ProximitySensors
- ledRed ()
- ledGreen ()
- ledYellow ()
- usbPowerPresent ()
- readBatteryMillivolts ()

CAPÍTULO III

IMPLEMENTACIÓN

3.1 Preliminares

El robot Zumo 32U4 es compatible con Arduino Leonardo, por lo tanto para descargar los programas desarrollados en el IDE de Arduino se debe seleccionar esta placa desde el menú Herramientas, ver figura 14.

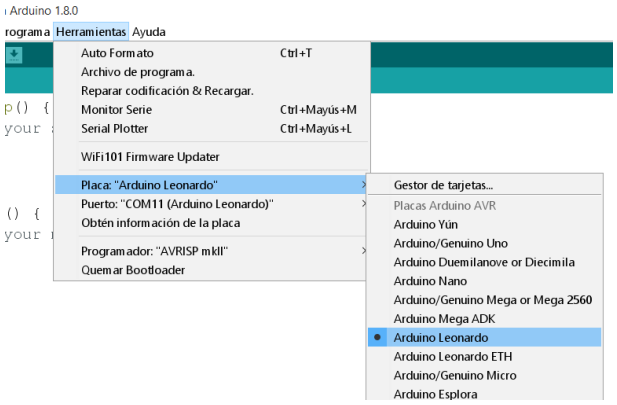

Figura 14 Selección de la placa compatible con robot Zumo

Para conectar el robot Zumo al computador es necesario un cable USB A aMicro-B, ver figura 15. Cuando se conecta por primera vez ya instalado el software de Arduino en el computador, se instalarán los controladores necesarios para que le asigne un puerto COM al robot.

Figura 15 Cable USB A a micro B

Terminada la instalación de los controladores del puerto para el robot, este aparece en el IDE de Arduino, seleccione desde el menú Herramientas, ver figura 16.

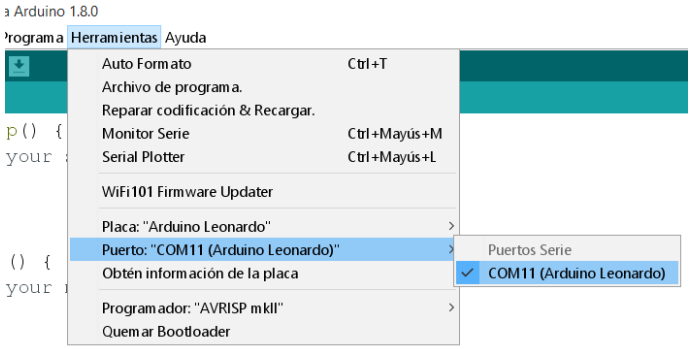

Figura 16 Selección del puerto asignado al robot Zumo

3.2 Comprobación del funcionamiento de dispositivos del robot Zumo

3.2.1 Parpadeo de los tres leds que están junto a los pulsadores

Los tres leds que están junto a los cuatro pulsadores etiquetados como A, B, C y Reset son de color Amarillo, Verde, Rojo, para comprobar el funcionamiento de los mismos se realiza en programa de parpadeen los tres al mismo tiempo cada 500 milisegundos.

Abra el software Arduino, cree un programa desde el menú Archivo con el nombre Parpadeo, desde el menú Programa, opción Incluir Librería, seleccione la librería Wire, esta librería permite la comunicación I2C con los dispositivos existentes en la placa de control del robot Zumo. La figura 17 muestra el procedimiento explicado.

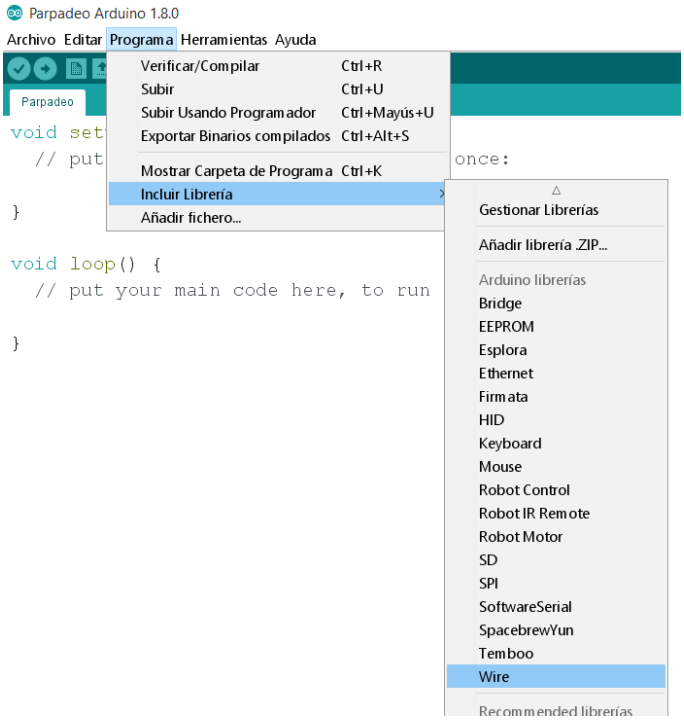

Figura 17 Selección de la librería Wire

De la misma manera agregue la librería Zumo32U4, esta contiene funciones que facilitan el control de los tres LED de usuario, LCD, pulsados. El programa con las dos librerías se observa en la figura 18.

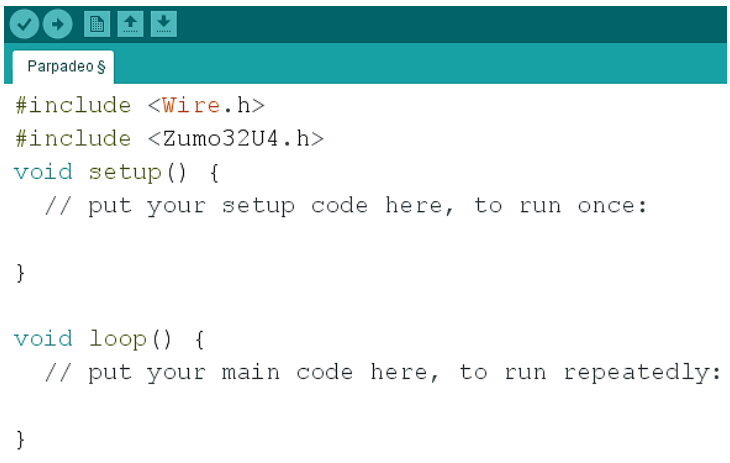

Figura 18 Librerías Wire y Zumo32U4 en el IDE de Arduino

En la figura 18 se observa que a continuación de las dos librerías esta la estructura setup( ), dentro de esta estructura se escriben líneas de código que se ejecutarán solo una vez cuando se enciende el robot o cuando se reinicia el mismo. Dentro de la estructura loop( ), se escribirá el código que

se estará ejecutando todo el tiempo. Estas dos estructuras se escriben automáticamente cuando se crea un nuevo programa.

Dentro de la estructura loop( ), escriba el código para encender los tres leds al mismo tiempo, las líneas de código son:

```
ledRed(1);
ledYellow(1);
ledGreen(1);
```

La línea de código para el retardo de 500 milisegundos es la siguiente:

```
delay(500);
```

Para apagar los tres leds se escribe lo siguiente:

```
ledRed(0);
ledYellow(0);
ledGreen(0);
```

Para agregar comentarios dentro de las líneas programación de Arduino se utiliza //.

Las instrucciones ledRed, ledYellow y ledGreen son parte de la librería Zumo32U4.h, se deben escribir sin modificar las letras mayúsculas o minúsculas.

El programa completo del parpadeo de los tres leds se muestra en la figura 19.

Para descargar el programa creado al robot Zumo, conecte al computador mediante el cable USB, seleccione la placa Arduino Leonardo y el puerto el que sea asignado en el computador, pulse el icono Subir que se encuentra bajo del menú Archivo, ver figura 20.

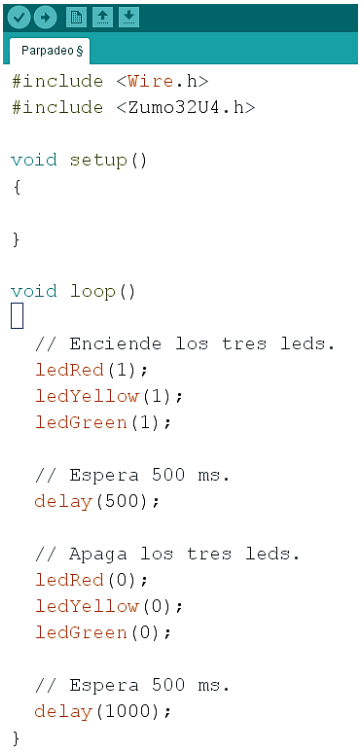

Figura 19 Programa del parpadeo de tres les de usuario

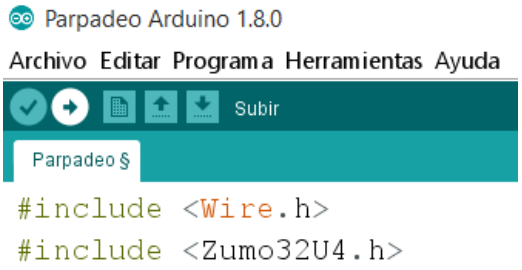

Figura 20 Icono para descargar programas al robot Zumo

Terminada la descarga del programa al robot observe que el led rojo, verde y amarillo están parpadeando los tres al mismo tiempo cada 500 milisegundos.

3.2.2 Visualización de texto en el LCD

El robot Zumo tiene incorporado una pantalla LCD de 8x2, esto significa 8 columnas y 2 filas, aquí se pueden mostrar textos y números de acuerdo a lo programado. Luego de las librerías Wire.h y Zumo32U4.h, se debe crear un objeto llamado lcd, puede ser cualquier nombre, este objeto servirá para ejecutar las tareas del LCD en el resto del programa, ver figura 21.

Figura 21 Creación del objeto lcd

La tabla 1 muestra las funciones empleadas en el manejo de un LCD

Tabla 1

Funciones para manejo de LCD

Función	Descripción
print( )	Imprime texto o números en pantalla
clear( )	Borra la pantalla
gotoXY(x,y)	Ubica en inicio de la impresión x→ representa columna y→ representa fila

El programa desarrollado en el IDE de Arduino se muestra en la figura 22.

Descargue el programa en el robot, observará en la primera fila del LCD la palabra Prueba que se imprime a partir de la segunda columna. En la segunda línea se muestra la palabra Pantalla.

```
pruebaLCD §
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD lcd;

void setup()
{

}

void loop()
{
  // Borra Pantalla
  lcd.clear();

  // Imprime en la primera fila
  lcd.print(" Prueba");

  // Ubicacion al inicio de la segunda linea
  lcd.gotoXY(0, 1);

  // Imprime en la segunda linea
  lcd.print("Pantalla");

  delay(1000);  //Retardo de 1 segundo
}
```

Figura 22 Programa para mostrar texto en el LCD

3.2.3 Programación de botones de usuario

Para comprobar el funcionamiento de los botones de usuario que se se encuentran en la placa principal de robot Zumo, se creó un programa que cuando se pulse el botón A se encienda el led amarillo, cuando se pulse el botón B se encienda el led verde y cuando se pulse el botón C se encienda el led rojo, cuando cualquier botón se encuentre sin pulsar los leds están apagados.

Es necesario crear tres objetos que son parte de la librería Zumo32U4.h que me permitan el acceso a los botones, estos se crear con las siguientes instrucciones:

Zumo32U4ButtonA buttonA;

Zumo32U4ButtonB buttonB;

Zumo32U4ButtonC buttonC;

Las siguientes líneas de código preguntan si el botón A esta presionado enciende el led amarillo caso contrario se apaga.

```
if (buttonA.isPressed())
  {
// Boton A presionadoenciende led amarillo
ledYellow(1);
}
else
  {
    //Boton A sin presionar apaga led amarillo
ledYellow(0);
  }
```

El programa completo para la comprobación de los botones de usuario es el siguiente:

```
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4ButtonA buttonA;
Zumo32U4ButtonB buttonB;
Zumo32U4ButtonC buttonC;

void setup() {
  // put your setup code here, to run once:

}

void loop() {
if (buttonA.isPressed())
{
    // Boton A presionado enciende led amarillo
ledYellow(1);
  }
else
  {
```

```
    //BotonA sin presionarapaga led amarillo
ledYellow(0);
  }

if (buttonB.isPressed())
{
// Boton B presionado enciende led amarillo
ledGreen(1);
  }
else
  {
//Boton B sin presionarapaga led amarillo
ledGreen(0);
  }

if (buttonC.isPressed())
{
    // Boton C presionado enciende led amarillo
ledRed(1);
  }
else
  {
    //Boton C sin presionar apaga led amarillo
ledRed(0);
  }
}
```

Descargue el programa en el robot Zumo, pulse el botón A y observe que se encienda el led amarillo, suelte el botón y el led se apagará, proceda de la misma manera con los dos botones restantes.

3.2.4 Generación de sonido

El robot Zumo trae incorporado un buzzer, el cual puede emitir sonidos a frecuencias determinadas y también reproducir notas musicales, el programa consiste en agregarle sonidos cuando se pulse cualquier boton de usuario, para ello se debe crear un objeto que permita cumplir con lo expuesto, el objeto creado se llama buzzer y se crea mediante la siguiente línea de código:

Zumo32U4Buzzer buzzer;

Mediante playFrecuency se generan los tonos por ejemplo la siguiente línea de programación genera un tono con una frecuencia de 400 Hz, por un tiempo de 200 ms y el máximo de volumen del buzzer que es 15.

buzzer.playFrequency(400, 200, 15);

El programa completo de la generación de tonos de acuerdo al botón pulsado es el siguiente:

```
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Buzzer buzzer;
Zumo32U4ButtonA buttonA;
Zumo32U4ButtonB buttonB;
Zumo32U4ButtonC buttonC;

void setup() {


}


void loop() {
```

```
if (buttonA.isPressed())
  {
ledYellow(1);
    //Genera tono de 800 Hz
buzzer.playFrequency(800, 200, 15);
  }
else
  {
ledYellow(0);
  }

if (buttonB.isPressed())
  {
ledGreen(1);
    //Genera tono de 400 Hz
buzzer.playFrequency(400, 200, 15);
  }
else
  {
ledGreen(0);
}

if (buttonC.isPressed())
  {
ledRed(1);
    //Genera tono de 1200 Hz
buzzer.playFrequency(1200, 200, 15);
  }
else
  {
ledRed(0);
```

```
  }

}
```

3.2.5 Sensores de proximidad

Para comprobar el funcionamiento de los sensores de proximidad frontales mostrados en la figura 7, cree un objeto llamado sensoresProx, con la siguiente línea de programación:

```
Zumo32U4ProximitySensors sensoresProx;
```

Este objeto es necesario inicializarlo, escriba la siguiente línea de programación en la estructura setup( ):

```
sensoresProx.initFrontSensor();
```

Para leer el valor de cada sensor infrarrojo sea este izquierdo o derecho es mediante las líneas siguientes:

```
sensoresProx.read();
uint8_t izquierda = sensoresProx.countsFrontWithLeftLeds();
uint8_t derecha = sensoresProx.countsFrontWithRightLeds();
```

Los resultados se observa en la pantalla LCD, el programa completo de la comprobación para los sensores infrarrojos frontales es el siguiente:

```
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD lcd;
Zumo32U4ProximitySensors sensoresProx;

voidsetup() {
```

```
// put your setup code here, to run once:
sensoresProx.initFrontSensor();
}

void loop() {
  // put your main code here, to run repeatedly:
sensoresProx.read();
uint8_tizquierda = sensoresProx.countsFrontWithLeftLeds();
uint8_tderecha = sensoresProx.countsFrontWithRightLeds();
lcd.gotoXY(0, 0);
lcd.print(izquierda);
lcd.gotoXY(0, 1);
lcd.print(derecha);
}
```

Descargue el programa en el robot y compruebe el funcionamiento de los sensores frontales.

Cuando el objeto esta fuera del rango de detección, más de 1,8 metros el valor es cero, a medida que se aproxima al sensor el valor incrementa hasta llegar a un máximo de 6, esta comprobación realice con la fuente de alimentación de las 6 pilas AA.

3.2.6 Prueba de motores

Para el control de los micromotoreductores de metal incluidos en el robot Zumo, es necesario crear un objeto llamado motores o cualquier otro nombre que es parte de la librería Zumo32U4.h, mediante la siguiente línea de código.

```
Zumo32U4Motors motores;
```

También para que los motores no inicien los movimientos en el momento de que termine de descargarse el programa es necesario una espera hasta

que se pulse un botón de usuario en este caso el botón A, la línea de código es la siguiente:

boton.waitForButton();

El programa consiste cuando pulse el botón A el robot de mueve hacia adelante por 2 segundos, luego de ese tiempo regresa con el movimiento hacia atrás por 2 segundos y se detiene, el programa completo para es el siguiente:

```
#include <Wire.h>
#include <Zumo32U4.h>
Zumo32U4ButtonA boton;
Zumo32U4Motors motores;

voidsetup() {
  //Espera hasta que el usuario presine el boton A
boton.waitForButton();
}

void loop() {
motores.setSpeeds(200, 200);
delay(2000);
motores.setSpeeds(-200, -200);
delay(2000);
motores.setSpeeds(0, 0);
boton.waitForButton();
}
```

3.3 Control de giro derecha e izquierda del robot

El programa cuando se pulsa el botón A, espera 2 segundos y gira el robot hacia la derecha, para esto solo funciona el motoreductor izquierdo hacia adelante, luego el moto reductor izquierdo rueda hacia atrás

provocando que el robot regresa a su posición inicial. Es ese instante se mueve solo el motor derecho hacia adelante produciendo un giro hacia la izquierda, luego el motor derecho rueda hacia atrás provocando que el robot regrese a su posición inicial, para que se repita de nuevo el ciclo es necesario pulsar el botón A nuevamente, el programa completo de lo expuesto es el siguiente:

```
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4Motors motors;
Zumo32U4ButtonA botonA;
voidsetup() {
botonA.waitForButton();
delay(2000);
}

void loop() {
//Rueda el motor izquierdo hacia adelante
ledYellow(1);
for (int speed = 0; speed <= 400; speed++)
  {
motors.setLeftSpeed(speed);
delay(2);
  }
for (int speed = 400; speed >= 0; speed--)
{
motors.setLeftSpeed(speed);
delay(2);
  }

  // Rueda el motor izquierdo hacia atras
ledYellow(0);
```

```
for (int speed = 0; speed >= -400; speed--)
  {
motors.setLeftSpeed(speed);
delay(2);
  }
for (int speed = -400; speed <= 0; speed++)
{
motors.setLeftSpeed(speed);
delay(2);
  }

  // Rueda el motor derecho hacia adelante
ledYellow(1);
for (int speed = 0; speed <= 400; speed++)
  {
motors.setRightSpeed(speed);
delay(2);
  }
for (int speed = 400; speed >= 0; speed--)
{
motors.setRightSpeed(speed);
delay(2);
  }

  // Rueda el motor derecho hacia atras
ledYellow(0);
for (int speed = 0; speed >= -400; speed--)
  {
motors.setRightSpeed(speed);
delay(2);
  }
for (int speed = -400; speed <= 0; speed++)
```

```
 {
motors.setRightSpeed(speed);
delay(2);
 }
botonA.waitForButton();
delay(2000);
}
```

3.4 Robot que localiza y sigue objetos

Este ejemplo utiliza el sensor de proximidad frontal del Zumo 32U4Frontal para localizar un robot o cualquier otro Objeto reflectante. Usando los motores para girar, escanea su alrededor es. Si detecta un objeto, enciende su LED amarillo. Cuando el objeto esta fuera de rango del sensor incrementa la velocidad de los motores al máximo hasta localizar algo.

El programa completo es el siguiente:

```
#include <Wire.h>
#include <Zumo32U4.h>

Zumo32U4LCD lcd;
Zumo32U4Motors motors;
Zumo32U4ProximitySensors proxSensors;
Zumo32U4ButtonA buttonA;

//Una lectura de sensores debe ser mayor o igual que este umbral
//para que el programa considere que el sensor ve un objeto.
const uint8_t sensorThreshold = 1;

//La velocidad máxima para accionar los motores mientras gira.
//400 es a toda velocidad.
const uint16_t turnSpeedMax = 400;
```

```
//La velocidad mínima para accionar los motores mientras gira.
const uint16_t turnSpeedMin = 100;

//La cantidad para disminuir la velocidad del motor durante
//cada ciclo cuando se ve un objeto.
const uint16_t deceleration = 10;

//La cantidad para incrementar la velocidad del motor durante
//cada ciclo cuando se ve un objeto.
const uint16_t acceleration = 10;

#define LEFT 0
#define RIGHT 1

//Almacena la última indicación de los sensores sobre la
//dirección a la que gira para hacer frente al objeto.
//Cuando no se ve ningún objeto, esta variable nos ayuda
//a hacer una buena suposición sobre la dirección a la que gira.
boolsenseDir = RIGHT;

//Es verdad el robot gira a la izquierda
//(en sentido contrario a las agujas del reloj).
boolturningLeft = false;

//Es verdad El robot gira a la derecha (sentido horario)
boolturningRight = false;

//Si el robot está girando, esta es la velocidad que utilizará.
uint16_t turnSpeed = turnSpeedMax;

//Tiempo en milisegundos, cuando se vio un objeto por última vez.
uint16_tlastTimeObjectSeen = 0;
```

```
void setup() {
proxSensors.initFrontSensor();

  //Es para que presione el botón A
lcd.clear();
lcd.print(F("Press A"));
buttonA.waitForButton();
lcd.clear();
}

voidturnRight()
{
motors.setSpeeds(turnSpeed, -turnSpeed);
turningLeft = false;
turningRight = true;
}

voidturnLeft()
{
motors.setSpeeds(-turnSpeed, turnSpeed);
turningLeft = true;
turningRight = false;
}

void stop()
{
motors.setSpeeds(0, 0);
turningLeft = false;
turningRight = false;
}
```

```
voidloop() {
//Lee el sensor de proximidad delantero y obtiene su valor de
//la izquierda (la cantidad de reflectancia detectada mientras
//utiliza los LED de la izquierda) y el valor de la derecha.
proxSensors.read();
uint8_tleftValue = proxSensors.countsFrontWithLeftLeds();
uint8_trightValue = proxSensors.countsFrontWithRightLeds();

//Determine si un objeto es visible o no.
boolobjectSeen = leftValue>= sensorThreshold || rightValue>=
sensorThreshold;

if (objectSeen)
 {
   //Un objeto es visible, inicia a desacelerar para
   //ayudar al robot a encontrar el objeto sin sobrepasar ni oscilar.
turnSpeed -= deceleration;
 }
else
 {
   //Un objeto no es visible, por lo que vamos a acelerar para ayudar
   //a encontrar el objeto antes.
turnSpeed += acceleration;
 }

//Restringir la velocidad de giro para que este
//entre turnSpeedMin y turnSpeedMax.
turnSpeed = constrain(turnSpeed, turnSpeedMin, turnSpeedMax);

if (objectSeen)
 {
```

```
//Objetovisto.
ledYellow(1);

lastTimeObjectSeen = millis();

boollastTurnRight = turnRight;

if (leftValue<rightValue)
   {
     //El valor ala derecha es mayor, por lo que el objeto está probablemente
     //más cerca de los LEDs correctos del robot, lo que significa que el
     //robot no está frente a él directamente. Gire a la derecha para
     //tratar de hacerlo más uniforme.
turnRight();
senseDir = RIGHT;
   }
else if (leftValue>rightValue)
{
     //El valor de la izquierda es mayor, así que gírelo a la izquierda.
turnLeft();
senseDir = LEFT;
   }
else
   {
     //Los valores son iguales, así que detenga los motores.
stop();
   }
}
else
 {
   //No se ve ningún objeto, así que solo sigue girando en la dirección
   //en que senso por última vez el objeto.
```

```
ledYellow(0);

if (senseDir == RIGHT)
   {
turnRight();
   }
else
   {
turnLeft();
   }

lcd.gotoXY(0, 0);
lcd.print(leftValue);
lcd.print(' ');
lcd.print(rightValue);
lcd.gotoXY(0, 1);
lcd.print(turningRight ? 'R' : (turningLeft ? 'L' : ' '));
lcd.print(' ');
lcd.print(turnSpeed);
lcd.print(' ');
lcd.print(' ');
}

}
```

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se implementó dos robots Zumo 32U4 uno con dos motoreductores de 75:1 HP y el otro con dos motoreductores de 50: HP compatibles con Arduino.
- El motoreductor de 50:1 HP tiene alta velocidad y par motor bajo mientras que el motoreductor de 75:1 tiene velocidad y par motor medio.
- Los dos robots Zumos implementados cumplen con las dimensiones para la competición en la categoría mini Zumo ya que mide 10 cm por lado y tiene un peso aproximado de 275 gr ya instaladas las baterías.
- Para que la programación del robot Zumo 32U4 sea más fácil es necesario descargarse en incluir en le IDE de Arduino la librería llamada Zumo32U4.h la misma que permite tener acceso a todos los dispositivos que conforman el robot.
- El robot Zumo 32U4 consiste de un chasis, la placa principal 32U4 en la que existe un microcontrolador integrado AVR ATmega32U4, una matriz de sensores frontales, controladores de motor, codificadores, sensores de proximidad, sensores de línea, zumbador, cuatro botones, un LCD, acelerómetro LSM303D y el giroscopio L3GD20H.
- Existe una etiqueta de color en el compartimiento de las pilas para identificar la relación del engranaje del micro motoreductor, la etiqueta de color azul identifica a uno de 75:1 HP y la de color verde que es un motoreductor de 50:1.

4.2 Recomendaciones

- Emplear como fuente de alimentación de los robots Zumo las pilas AA de níquel hidruro metálico (Ni-Mh) recargables como mínimo de 2200 mAh.
- Seleccionar en el IDE de Arduino la placa de Arduino Leonardo que es compatible con el microcontrolador Atmega 32U4 que viene en el robot Zumo.
- No cambiar a la posición ON el interruptor de encendido de la placa principal del robot cuando se está descargando programas desde el computador.
- Entregar al club de robótica estos dispositivos para que continúen con la investigación de todas las funciones que ofrecen estos robots.

GLOSARIO DE TÉRMINOS.

AVR	Familia de microcontroladores de la empresa Atmel
CAD	Diseño Asistido por Computador
CPR	Cuentas por Revolución
IDE	Entorno de Desarrollo Integrado
IMU	Unidad de Medición Inercial
IR	Infrarrojo
JIRA	Asociación Japonesa de Robótica Industrial
LCD	Display de Cristal Líquido
LED	Diodo Emisor de Luz
PCB	Placa de Circuito Impreso
PWM	Modulación por Ancho de Pulso
RIA	Instituto de Robótica de Norteamérica
USB	Bus Serial Universal

REFERENCIA BIBLIOGRAFÍA.

Corporation, P. (2015). *Pololu Zumo 32U4 Robot User's*. Las Vegas, USA.

López, L. (Septiembre de 2013). *Robótica Educativa*. Quito, Pichincha, Ecaudor.

Romero Costas, M. (2012). Robótica entra al mundo de la inteligencia artificial. Buenos Aires, Argentina: Edu.ar S.E.

Torrente, O. (2013). *Arduino. Curso práctico de formación.* México: Alfaomega Grupo Editor, S.A.

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

ANEXOS