UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:

INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

TEMA:

IMPLEMENTACIÓN DE UN PROTOTIPO DE ROBOT BÍPEDO CON TECNOLOGÍA XBEE PARA LA DETECCIÓN DE OBSTÁCULOS PREDETERMINADOS.

AUTOR:

Carlos Xavier Toapanta Paucar

TUTOR:

MSc. Elizabeth Patricia Carrillo Armendariz

QUITO - ECUADOR

AÑO: 2019

UNIVERSIDAD TECNOLÓGICA ISRAEL

DECLARACIÓN DE AUTORÍA

Yo, Carlos Xavier Toapanta Paucar, declaro que los resultados obtenidos en la investigación que presento como informe final, previo a la obtención del título de Ingeniero en Electrónica Digital y Telecomunicaciones, son absolutamente originales, auténticos y de mi autoría; que el presente trabajo no ha sido previamente presentado para ningún grado profesional o académico; y que he consultado las referencias bibliográficas que se incluyen en este documento.

En tal virtud, expreso que el contenido, las conclusiones y los efectos legales y académicos que se desprenden del presente trabajo es de exclusiva responsabilidad del autor.

.....

Carlos Xavier Toapanta Paucar

CI.:1723715163

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR:

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación "IMPLEMENTACIÓN DE UN PROTOTIPO DE ROBOT BÍPEDO CON TECNOLOGÍA XBEE PARA LA DETECCIÓN DE OBSTÁCULOS PREDETERMINADOS.", presentado por el Sr. Carlos Xavier Toapanta Paucar, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D.M. 16 agosto del 2019

TUTOR

.....

MSc. Elizabeth Patricia Carrillo Armendariz

AGRADECIMIENTOS

A mi Dios por permitirme seguir adémate y por guiarme en el sendero del bien. A mis padres por apoyarme en mi formación académica y por enseñarme a ser perseverante en las adversidades. De igual manera mi agradecimiento a los Ingenieros de la carrera de Electrónica de la Universidad Israel.

Carlos Xavier

DEDICATORIA

A mi Dios por permitirme llegar a culminar esta etapa en la vida, a mis padres por ser los pilares principales en mi formación académica y humana, a mi amada esposa por su apoyo y paciencia, a mi hijo por ser mi motivación para superarme, a mi hermana por estar conmigo en todo momento, a mis abuelitos por su cariño desde donde estén siempre los tendré en mi pensamiento, les dedico la realización de este proyecto.

Carlos Xavier

ÍNDICE

ÍNDICE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Hoy en día el adelanto tecnológico continúa creciendo constantemente, esto se debe al desarrollo de la robótica a nivel mundial. Este es el caso de los robots bípedos los cuales han sido diseñado para interactuar con su entorno mediante sensores y además brindar diversos tipos de asistencia en diferentes áreas entre estas: la exploración de lugares de difícil acceso y en la educación para promover la creatividad de los estudiantes. Todo esto ha llevado a plantear el presente trabajo el cual tiene como objetivo principal la implementación un prototipo de robot bípedo con tecnología Xbee para la detección de obstáculos predeterminados. Este proyecto ha sido desarrollado para dar a conocer un instructivo sobre la elaboración de un robot bípedo. El cual podrá ser utilizado para incentivar la robótica educativa como una herramienta de motivación para el aprendizaje en los estudiantes o personas interesadas por el estudio de la robótica. Para resolver este problema se utilizó la metodología planteada por Angulo, J. (1986). Esta metodología consiste en 7 etapas, las cuales han sido aplicadas para la elaboración de la guía de ensamblaje. Esto ha permitido obtener el siguiente resultado: se ha elaborado un prototipo bípedo el cual funciona correctamente y que en base a su construcción se ha elaborado un instructivo el cual podrá ser utilizado por las personas para el desarrollo de un robot. En conclusión, este proyecto es un aporte hacia la sociedad porque incentiva a las personas a investigar sobre el desarrollo tecnológico para que en un futuro sean capaces de aplicar estos conocimientos en un proyecto de beneficio para la sociedad.

Palabras clave: robótica, tecnología, diseño electrónico, robot bípedo, interfaz inalámbrica.

ABSTRACT

Today, technological progress continues to grow steadily, this is due to the development of robotics worldwide. This is the case of bipedal robots which have been designed to interact with their environment through sensors and also provide various types of assistance in different areas among these: the exploration of hard-to-reach places and in education to promote the creativity of students. All this has led to the present work which has as main objective the implementation of a biped robot prototype with Xbee technology for the detection of predetermined obstacles. This project has been developed to publicize an instruction on the development of a biped robot. Which can be used to encourage educational robotics as a motivational tool for learning in students or people interested in the study of robotics. To solve this problem, the methodology proposed by Angulo, J. (1986) was used. This methodology consists of 7 stages, which have been applied for the preparation of the assembly guide. This has allowed obtaining the following result: a biped prototype has been developed which works correctly and based on its construction an instructional has been developed which can be used by people for the development of a robot. In conclusion, this project is a contribution to society because it encourages people to investigate technological development so that in the future they will be able to apply this knowledge in a project that benefits society.

Keywords: robotics, technology, electronic design, biped robot, wireless interface.

INTRODUCCIÓN

Antecedentes de la situación objeto de estudio

El estudio de la robótica se ha caracterizado por reproducir los movimientos de los seres vivos, este es el caso de los robots bípedos los cuales para su construcción presentan un cierto grado de complejidad debido al factor equilibrio al momento de desplazarse. (Morillo & Portero, 2014). Entre los robots bípedos más conocidos están los desarrollados por la empresa de origen Japonés Honda con el robot ASIMO (*Advanced Step in Innovative Movile*), la evolución de este robot inicio en el año de 1986 el cual le llevo a esta empresa 14 años de investigación. (Herrera, 2009).

En el Ecuador se han realizado trabajos de investigación referentes a los robots bípedos. Ente ellos se pude mencionar el trabajo realizado en la Escuela politécnica Nacional en el año 2009 por Marco Antonio Herrera Garzón, con el tema: Ensamblaje y control de una plataforma robótica bípeda mediante un PC. En este proyecto de investigación se utiliza un microcontrolador ATmega8, un sensor ultrasónico y un módulo de radiofrecuencia para controlar los movimientos del robot de forma inalámbrica. (Herrera, 2009).

También se puede mencionar el trabajo realizado en la Universidad de Guayaquil en el año 2018 por Martínez Cruz Darwin Avelino, con el tema: Robonoide bípedo autónomo realizado en plataforma Arduino. Este proyecto de investigación fue elaborado mediante una placa Arduino Nano y para el diseño de la estructura se utilizó una impresora 3D. Además presenta una guía en la que se explica paso a paso el desarrollo del Robonoide. (Martínez, 2018)

Con referencia a lo anterior mencionado se debe destacar el desarrollo que ha tenido desde sus inicios hasta la actualidad. Es por esto que actualmente se está empleando la robótica en el ámbito educativo ya que desarrolla competencias en los estudiantes como la creación y la creatividad. Por lo tanto, el desarrollo de este proyecto permitirá a los estudiantes llenar los vacíos en la parte teórica y en la parte práctica.

Planteamiento del problema

En la actualidad el avance tecnológico en el campo de la robótica ha crecido considerablemente, de tal manera que ha permitido cambiar algunos aspectos de la vida cotidiana de las personas, por ejemplo: el ensamblaje de automóviles, prótesis robóticas y para la exploración de lugares de difícil acceso. Esto ha sido posible debido a la aplicación de la robótica en escuelas, hogares y en hospitales.

En el área científica se ha visto la necesidad de aplicar otras tecnologías como la inteligencia artificial y el uso de sensores, para la elaboración de prototipos que sean capaces de interactuar con el entorno humano de forma independiente, todo esto con la finalidad de poner a la robótica al beneficio de las personas en actividades de alto riesgo.

La elaboración de robots representa un conocimiento avanzado sobre diseño electrónico y programación de algoritmos, los cuales por lo general son impartidos en instituciones de educación superior. Sin embargo, existen personas que se encuentran atraídos por el estudio de la robótica, los cuales no disponen de los conocimientos suficientes para elaborar un determinado robot y como consecuencia presentan diversos vacíos tanto en la parte teórica como en la parte práctica.

En el desarrollo de un prototipo robótico es necesario conocer el funcionamiento de los componentes electrónicos lo cual permitirá elaborar un robot que funcione correctamente ya que de lo contrario los elementos electrónicos podrían dañarse por no ser utilizados adecuadamente. Este inconveniente puede generar un gasto económico adicional y a su vez se extendería el tiempo de construcción. Por este motivo los estudiantes presentan dificultades en el proceso de elaboración de un robot.

Justificación

En la actualidad el estudio de la robótica puede aplicarse para mejorar la enseñanza aprendizaje en los estudiantes, ya que través del uso de las herramientas tecnológicas el estudiante desarrolla capacidades como la creatividad y les permite dar soluciones a diversos problemas.

La tarea de incentivar a una persona a que sienta ganas de estudiar puede ser de cierta forma muy complicado, sin embargo, esto puede cambiar mediante el uso de la robótica educativa ya que esta disciplina puede ser impartida para desarrollar competencias como la creatividad y el interés por la investigación. El presente trabajo consiste en implementar un prototipo de robot bípedo con tecnología Xbee para la detección de obstáculos predeterminados, y ha sido desarrollado para dar a conocer un instructivo sobre la elaboración de un robot bípedo. Este proyecto está enfocado en aportar a la sociedad una estrategia educativa innovadora para el aprendizaje de las personas mediante el uso de la robótica educativa. Esto permitirá fortalecer una parte del aprendizaje en donde se interactúa. De este modo se busca despertar en los estudiantes, el entusiasmo por el desarrollo de habilidades que les permitan la construcción de un saber fundamentado en la tecnología

Objetivo general

- Implementar un prototipo de robot bípedo con tecnología Xbee para la detección de obstáculos predeterminados.

Objetivos específicos

- Establecer los parámetros de funcionamiento del robot, para que su desplazamiento sea controlado inalámbricamente desde una computadora.
- Definir los elementos electrónicos y mecánicos para determinar los materiales a utilizar en la construcción.
- Diseñar un prototipo de robot bípedo, basado en tecnología Xbee.
- Desarrollar el algoritmo de control y funcionamiento del robot.
- Construir un robot con parámetros de estabilidad y movilidad para que sea capaz de moverse sobre una superficie plana.
- Realizar las pruebas de funcionamiento para comprobar las funciones programadas en el robot.

Alcance

El alcance de este proyecto de titulación está definido de acuerdo al objetivo planteado, de la siguiente forma: Implementar un prototipo de robot bípedo con tecnología Xbee para la detección de obstáculos predeterminados.

Su funcionamiento consiste en utilizar la interfaz de usuario y mediante esta controlar el desplazamiento del robot para que este esquive obstáculos previamente determinados. La interfaz de usuario permitirá controlar al robot mediante el control manual y el control automático.

El control manual se utiliza para dirigir la dirección del desplazamiento y para esto se utiliza los botones: adelante, izquierda y derecha. En cambio, con el modo de control automático el robot se desplaza de forma autónoma y para esto utiliza el sensor de detección. La función de este sensor es detectar obstáculos dentro de un rango predeterminado para que el robot no choque con él con el obstáculo.

Estos obstáculos deben presentar las siguientes características para que sean detectados por el sensor: los objetos deben encontrarse a la misma altura del sensor, poseer un ancho superior a los 10 centímetros, encontrarse dentro de una distancia de 25 centímetros, sólidos, estáticos, que no sean transparentes y que se encuentre ubicado en un lugar abierto con claridad.

El algoritmo de la placa electrónica será desarrollado en el software Arduino, en cambio el algoritmo de la interfaz de usuario será diseñado en el software Visual Studio. Para enlazar la comunicación entre el robot y la interfaz de usuario se utiliza módulos de comunicación Xbee.

Este robot está construido para desplazarse sobre una superficie plana que tenga las siguientes características: contextura sólida, estática, anti deslizante y sin ángulo de inclinación. Además, este documento cuenta con: un manual de usuario (Anexo 1) y un manual técnico (Anexo 2).

Descripción de los capítulos

En el capítulo 1 se presenta el Marco Teórico, en donde se describe el funcionamiento de los robots con movilidad, haciendo un énfasis en los robots bípedos y una breve explicación de la locomoción humana. Del mismo modo los componentes del objeto de estudio, entre estos: el microcontrolador, tarjeta Arduino, sensor de distancia y los servomotores.

En el capítulo 2 se describen los aspectos metodológicos y prácticos que se utilizaron para la Implementación del robot bípedo con tecnología Xbee. También se describe el tipo de investigación realizada, las técnicas e instrumentos utilizados y la metodología seleccionada.

En el capítulo 3 se documenta la Propuesta de diseño, en el cual se establece los parámetros para la elaboración del proyecto de titulación los cuales están establecidos de la siguiente forma: características del robot, materiales a utilizar, diseño de circuitos, y los diagramas de flujo para la estructuración de los algoritmos de programación.

En el capítulo 4 se describe el proceso de construcción del robot bípedo, se realiza pruebas para verificar el correcto funcionamiento del prototipo. Para esto se verifica el desplazamiento del robot en una superficie plana, se verifica el control mediante la interfaz de usuario y también se comprueba el funcionamiento del sensor.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

En este capítulo se da a conocer el concepto de la robótica y sus aplicaciones dentro de la sociedad, también se hace un énfasis en los sistemas robóticos con desplazamiento bípedo. Además, se describe el funcionamiento de los componentes electrónicos a utilizar en este proyecto.

1.1 Robótica

La robótica es una ciencia que involucra varias disciplinas como la informática, la electrónica y la mecánica. Esto ha permitido definir al robot como un sistema multifuncional y reprogramable capaz de realizar trabajos sencillos hasta aquellos que son riesgosos para el ser humano de forma autónoma. Básicamente está constituido por una estructura mecánica y un sistema de control que le permiten realizar tareas en diferentes áreas. En la figura 1.1 se muestra los robots construidos por la Empresa Honda.

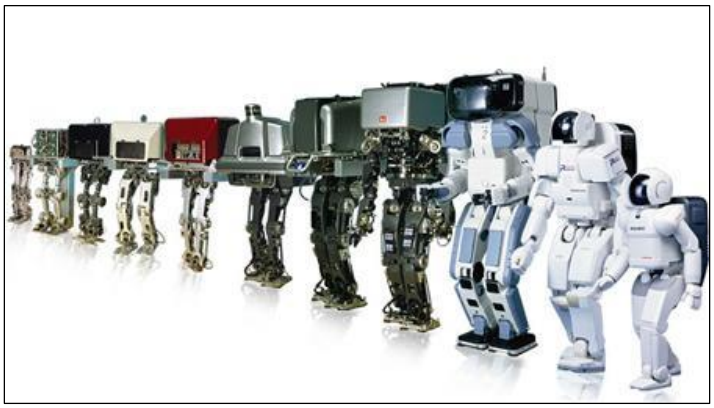


Figura 1.1: Los robots de Honda
Fuente: (Intelligence Artificielle, s.f.)

En la actualidad el estudio de la robótica ha despertado gran interés en la comunidad científica, por tal motivo se han diseñado robos especialmente para la investigación y para el desarrollo tecnológico. Mediante el estudio de la robótica se busca mejorar continuamente los sistemas de control que permitan a un robot interactuar y comunicarse de forma sencilla con las personas. (Cela, 2012)

1.2 Robótica móvil

La robótica móvil en el transcurso de los últimos años ha evolucionado con los avances tecnológicos esto ha permitido desarrollar robots con múltiples aplicaciones en donde se aplica la autonomía, la cual hace referencia a la capacidad del robot para desempeñarse en su área de trabajo sin el control externo de una persona (Durán & Granja, 2010). En la actualidad estos robots son excesivamente costosos y sus funciones en el campo de trabajo son muy limitadas. Es por esto que el estudio de la robótica se ha centrado en construir robots móviles con mayores capacidades a un costo menor. Estos robots están constituidos por diferentes capacidades entre estas: el desplazamiento o locomoción, y la interrelación con su entorno. En donde estas capacidades son controladas desde el microcontrolador. (Ramírez & Reyes, 2015)

1.2.1 Robots móviles con ruedas

Es aquel robot que está estructurado por ruedas, las cuales son controladas mediante un motor de corriente continua, de esta manera el robot consigue desplazarse en cualquier dirección sobre una superficie. Su principal ventaja es conseguir velocidades altas, además este tipo de robot puede presentar desventajas en terrenos irregulares ya que puede patinar. (Ramírez & Reyes, 2015). En la figura 1.2 se muestra un robot construido con ruedas.

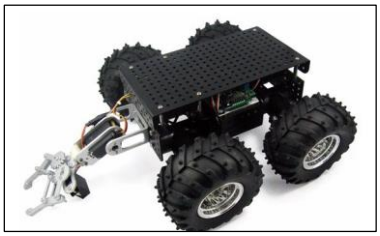


Figura 1.2: Wild Thumper
Fuente: (GoTronic, s.f.)

1.2.2 Locomoción mediante patas

Este tipo de robot imita las distintas formas de desplazamiento tanto de animales como la de los seres humanos. Se destacan por ser utilizados en diferentes aplicaciones y en varias superficies, pero su principal desventaja es la velocidad en la que se desplaza, la cual es lenta en comparación al robot con ruedas. En el diseño de estos robots con patas se debe considerar ciertos aspectos como: posición, velocidad y equilibrio, puesto que al estar constituidos por patas su complejidad aumenta debido a los movimientos a coordinar. Como por ejemplo los sistemas que se basan en el desplazamiento bípedo en donde su complejidad aumenta al momento de mantener la estabilidad. Sin embargo, en un sistema cuadrúpedo su complejidad es menor debido al número de patas que posee, las cuales le proporcionan mayor estabilidad al momento de desplazarse. (Cruz, 2017). En la figura 1.3 se muestra un robot cuadrúpedo.

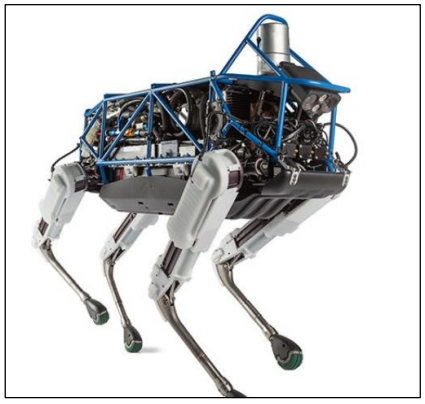


Figura 1.3: Robot Spot
Fuente: (Boston Dynamics, s.f.)

1.3 Estudio de la locomoción bípeda

La locomoción bípeda se basa en el desplazamiento del ser humano y en las posibles aplicaciones en sistemas controlados o autónomos (Herrera, 2009). El desplazamiento bípedo consiste en enviar órdenes desde el sistema de control hacia las partes que conforman las extremidades inferiores de manera que mantengan el equilibrio (Herrera, 2009). Este proceso también es conocido como ciclo de marcha, en donde se analiza los pasos para generar la caminata.

1.3.1 Ciclo de marcha

El ciclo de la marcha se refiere a un punto inicial y a un punto final dentro de la caminata, en donde el punto inicial se efectúa cuando el pie toca una superficie y termina cuando el mismo pie tocar nuevamente la superficie. Como se muestra en la figura 1.4.

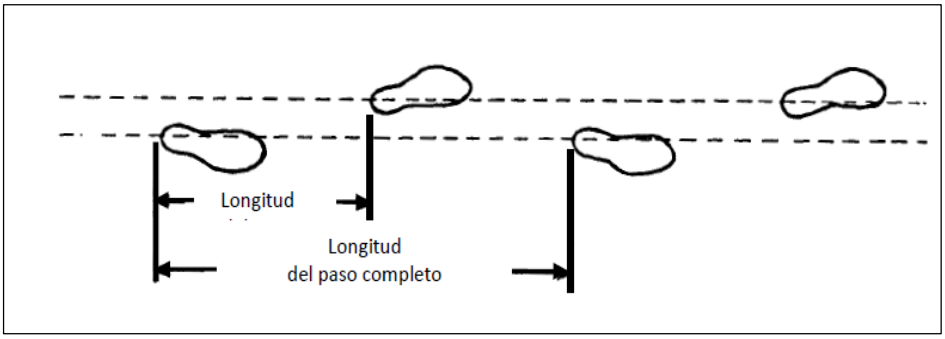


Figura 1.4: Longitud del paso
Fuente: (Guzmán , 2010)

Dentro de la caminata se puede apreciar el proceso de estabilidad el cual inicia con la fase de apoyo cuando el pie hace contacto con el suelo y a la fase de balanceo cuando el pie no toca el suelo. La locomoción bípeda se basa básicamente en imitar del desplazamiento motriz del ser humano en donde es importante mantener el equilibrio en cada paso. En este proceso se deben desarrollar movimientos de forma sincronizada como se observa en la figura 1.5, en cada una de las partes del cuerpo tanto en piernas como en brazos. citado por (Morillo & Portero, 2014).

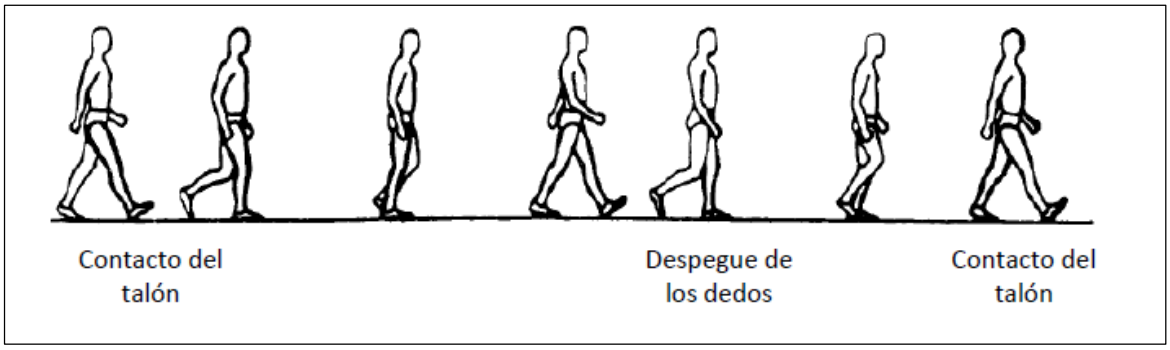


Figura 1.5: Ciclo del caminado humano
Fuente: (Guzmán , 2010)

Entre los aspectos de la caminata es importante conocer cómo se genera la misma, es por esto que en la figura 1.6 se puede observar esta acción, y para analizar este proceso se lo divide en cinco literales.

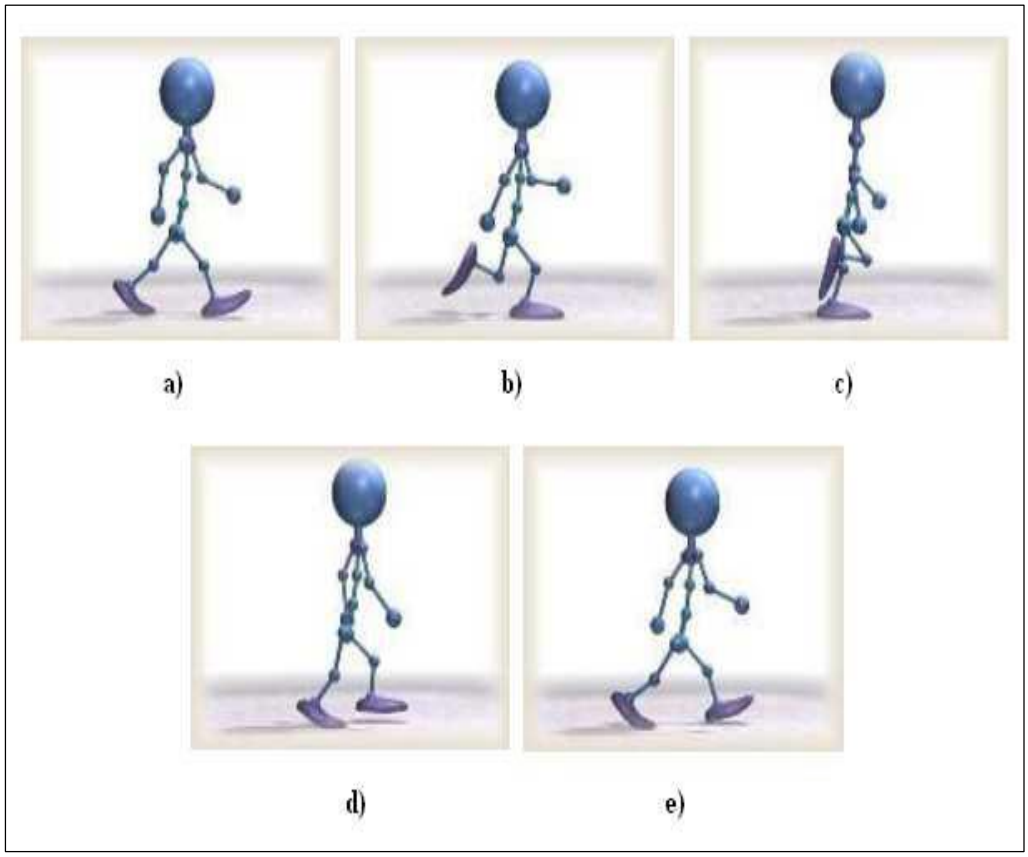


Figura 1.6: Caminata humana
Fuente: (Herrera, 2009)

A continuación, se analiza el desplazamiento del ser humano: a) se puede observar como inicia el proceso de la caminata, en donde una de las piernas se desplaza hacia adelante y por ende también el peso del cuerpo, b) todo el peso cae sobre la rodilla y esta se dobla para absorber el choque, c) la rodilla se endereza haciendo que el cuerpo se desplace hacia delante, mientras que el pie libre pasa a un lado del pie de apoyo, d) el cuerpo cae hacia delante hasta que el pie libre alcance a tocar el suelo. (Herrera, 2009), e) el pie libre toca el piso completamente, a esto se le considera como la mitad del ciclo de caminado. (Herrera, 2009). Este proceso es completado con el segundo ciclo el cual es el reflejo del primero. (Herrera, 2009)

1.4 Robot bípedo

Un robot bípedo es aquel que dispone de dos piernas para su desplazamiento. En su mayoría intentan imitar el sistema motriz de hombre para desplazarse en su entorno sin perder el equilibrio. (Herrera, 2009)

1.4.1 Elementos fundamentales

El robot bípedo es una estructura mecánica que cuenta con segmentos y articulaciones. El cual mediante un microcontrolador ejecuta los movimientos programados. A continuación, los elementos que lo constituyen.

- **Controlador**

Es la parte central que controla al robot o también conocido como microcontrolador. Se encarga de gobernar el funcionamiento del sistema. (Parra, 2015). Este dispositivo es el encargado de recibir la información de los sensores para efectúa las órdenes establecidas mediante los actuadores. (Parra, 2015)

- **Actuadores**

Los actuadores permiten que los segmentos del robot o partes rígidas, alcancen un estado determinado, en donde el microcontrolador es el encargado de imponer esta orden o instrucción. (Rodríguez & Berenguel, 2012)

- **Sistema sensorial**

Hacen referencia a los sensores, los cuales permiten detectar el entorno del robot para realizar una acción determinada mediante la programación. (Rodríguez & Berenguel, 2012)

- **Articulación**

La articulación es la unión entre dos piezas rígidas o también conocidas como segmentos como se observa en la figura 1.7, en donde se genera el movimiento entre ellas. La articulación más empleada es la de tipo rotacional, la cual permite un movimiento de rotación entre un segmento en relación al otro. (Rodríguez & Berenguel, 2012).

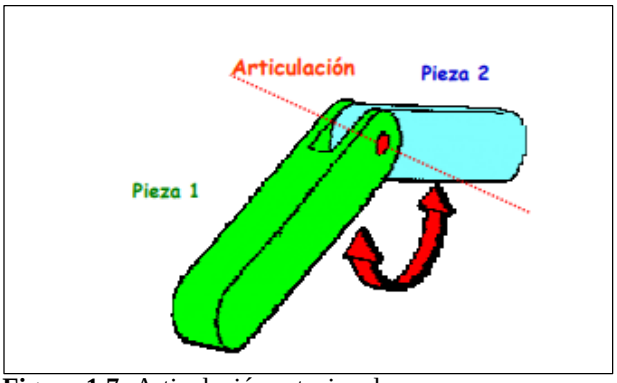


Figura 1.7: Articulación rotacional
Fuente: (Rodríguez & Berenguel, 2012)

- **Grados de libertad**

Los grados de libertad (GDL) hacen referencia a los movimientos que realiza un cuerpo en un espacio determinado. Los grados de libertad (GDL) en una articulación permiten realizar varios movimientos, en donde una pieza gira en varias direcciones con respecto a otra. (Rodríguez & Berenguel, 2012). En la figura 1.8 se muestra tres tipos de articulaciones que corresponden a diferentes grados de libertad.

Figura 1.8: Grados de libertad
Fuente: (Rodríguez & Berenguel, 2012)

- **Ecuación para determinar los grados de libertad**

Los grados de libertad en un sistema mecánico constituyen el número de movimientos relativos los cuales son generados entre los cuerpos rígidos. Para calcular los grados de libertad se utiliza la ecuación de Grübler. (Granja, 2014), la cual se muestra a continuación:

$$GDL = 3\,(n - 1) - 2j1 - j2 \qquad \text{Ecuación 1}$$

GDL= grados de libertad

n = número de eslabones

j1 = número de uniones de 1 GDL

j2 = número de uniones de 2 GDL

- **Ecuación para determinar el torque**

Para el estudio del torque de los servomotores se debe calcular el escenario de mayor momento. (Cevallos, 2019), para esto de utiliza la fórmula que se muestra a continuación:

$$Momento = Fmotor * \frac{La}{2} + Fmotor * \left(La + \frac{Lc}{2}\right) + \frac{Fcuerpo}{4} * (La + Lc * Lm) \qquad \text{Ecuación 2}$$

La = Medida de servo a servo en el tobillo

Lc = Medida de servo a servo en la rodilla

Lmx = Mitad de la estructura central en coordenada “x”

Lmy = Mitad de la medida de estructura central en coordenada “y”

Fmotor = Peso servomotor

Fcuerpo = Peso de cuerpo de robot

Para calcular la hipotenusa de la estructura central se utiliza la siguiente ecuación:

$$Lm = \sqrt{Lmx^2 + Lmy^2} \qquad \text{Ecuación 3}$$

Lm = Hipotenusa de la estructura central

Lmx = Mitad de la estructura central en coordenada “x”

Lmy = Mitad de la estructura central en coordenada “y”

1.4.2 Control Automático

El control automático de proceso es producto de una evolución consecuente al uso de las técnicas de medición y control. El control automático se usa fundamentalmente para reducir costos de producción, esto permite recuperar la inversión realizada al adquirir los equipos de control. Además, permite reducir la mano de obra pasiva y eliminar errores en la elaboración del producto final. En consecuencia, el control automático se genera debido a una acción la cual a su vez genera una reacción, todo esto sin la intervención humana. (Escalona & Morillo, 2017).

- **Clasificación del sistema de control**

Los sistemas de control se clasifican en sistemas de lazo abierto y lazo cerrado. Un sistema de lazo abierto es aquel en que la acción de control es independiente a la señal de la salida. Este sistema se caracteriza por ejecutar una acción determinada mediante la calibración preestablecida. En cambio, el sistema de control de lazo cerrado es aquel en el que la acción de control depende de la señal de salida. Este sistema es conocido como sistema de control por realimentación. (Escalona & Morillo, 2017). En la figura 1.9 se muestra los lazos de control.

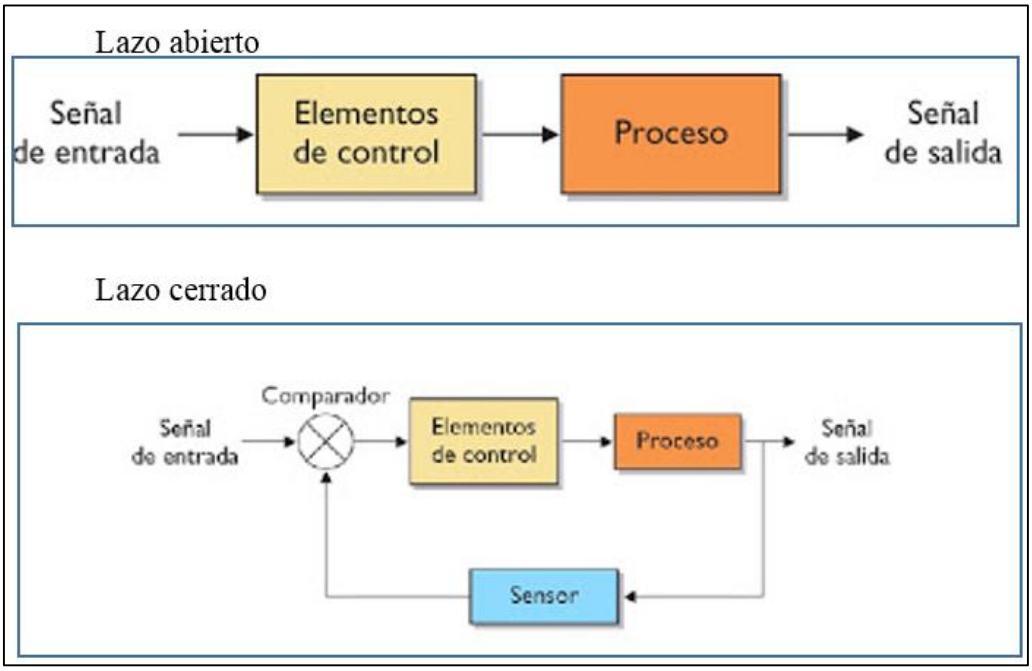


Figura 1.9: Control automático cerrado
Fuente: (Tecnologia, 2016)

1.4.3 Robot Troody

Es un robot bípedo que ha sido diseñado con forma de dinosaurio como se muestra en la figura 1.10. La forma física de Troody está constituida por una cabeza, una cadera y dos piernas. Este robot fue construido por Peter Dilworth en el Instituto de Tecnología de Massachusetts (MIT). Las dimensiones del robot Troody fueron asemejadas al fósil del dinosaurio troodon. Además, cuenta con 16 grados de libertad (GDL) y un giroscopio. (Hizzook, 2008).

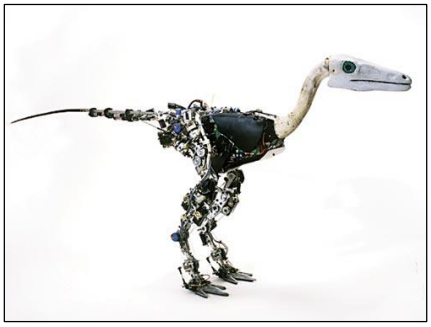


Figura 1.10: Robot Troody
Fuente: (Wired, 2004)

1.4.4 Robot BRAT

La robótica bípeda ha despertado gran interés en la comunidad científica, hoy en día existen algunos robots diseñados especialmente para investigación y desarrollo. Tal es el caso del robot BRAT (Robot Articulado de Trasporte Bípedo), ilustrado en la figura 1.11, este robot está diseñado con 6 grados de libertad, 2 en su cadera, 2 en sus rodillas y 2 en sus tobillos. Se caracteriza por tener estabilidad al momento de desplazarse (Herrera, 2009).

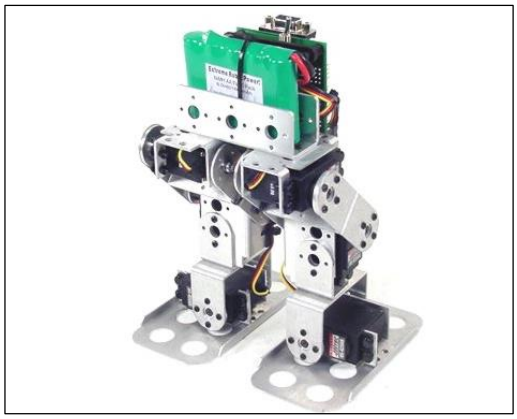


Figura 1.11: Robot BRAT
Fuente: (Lynxmotion, 2018)

1.5 Robot humanoide

El ser humano se ha esforzado por comprender los sistemas y mecanismos que componen el cuerpo humano, como el control muscular, sistema nervioso, los mecanismos de aprendizaje y reproducir la forma de caminar. En los últimos años se han podido desarrollar robots humanoides que imitan la estructura del cuerpo humano los cuales se han diseñado para tareas principales como la asistencia a personas y para la repuesta a desastres (Asano, Okada, & Inaba, 2017)

En el mercado tecnológico se conoce de varios proyectos que se han dado a conocer rápidamente por sus nuevos avances dentro de la robótica. Estos robots han sido desarrollados tras varios años de trabajo, con el único propósito de posicionar sus productos en el mercado. A continuación, se describe alguna de las características de estos robots humanoides (Parra, 2015).

1.5.1 Robot ASIMO

El robot ASIMO *(Advanced Step in Innovative Mobility*), mostrado en la figura 1.12, fue creado en el año 2000 por la compañía japonesa "Honda". Fue construido para asistir a las personas y realizar tareas domésticas. En el 2013 ASIMO ha sido mejorado con tecnologías de nueva generación, esto le permite reconocer objetos, imágenes, gestos, reconocimiento facial, reconocimiento de voz y además analizar su entorno para desplazarse por él interactuando con seres humanos.

Figura 1.12: Robot ASIMO
Fuente: (Flickr, 2010)

El robot ASIMO está diseñado con una altura de 120 centímetros (cm), cuenta con un peso de 52 kilogramos (kg), 26 grados de libertad (GDL) y tiene un tiempo de funcionamiento de forma autónoma con baterías de 30 minutos. Su principal desventaja es que no es capaz de levantarse cuando este cae al piso. (Lezama & Sklar, 2004)

1.5.2 Robot Kengoro

Este tipo de robot es considerado como humanoide, porque ha sido diseñado en base al cuerpo humano con las mismas proporciones corporales y con la misma estructura musculoesquelética, como se observa en la figura 1.13. Kengoro está compuesto por 114 grados de libertad (GDL) y 116 músculos actuadores, esto le permite realizar toda clase de ejercicios como: flexiones de pecho, abdominales, estiramientos, etc. (Asano, Okada, & Inaba, 2017).

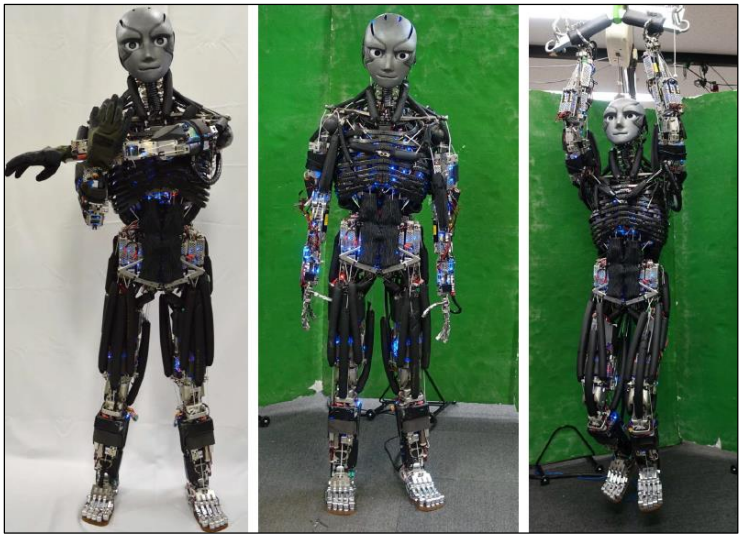


Figura 1.13: Robot Kengoro
Fuente: (Asano, Okada, & Inaba, 2017)

1.6 Arduino Mega

Es una tarjeta electrónica que se utiliza para el desarrollo de entornos interactivos de código abierto y fácil de utilizar. A diferencia de otras plataformas esta tarjeta puede ser ejecutada en varias plataformas como: Windows, Mac OS y Linux. Está constituido por un microcontrolador ATmega 2560 y puede ser programado desde cualquier computador y por su fácil manejo es muy utilizado en la robótica y la domótica. (Thayer, 2017).

A continuación, se muestra en la figura 1.14 la tarjeta Arduino Mega.

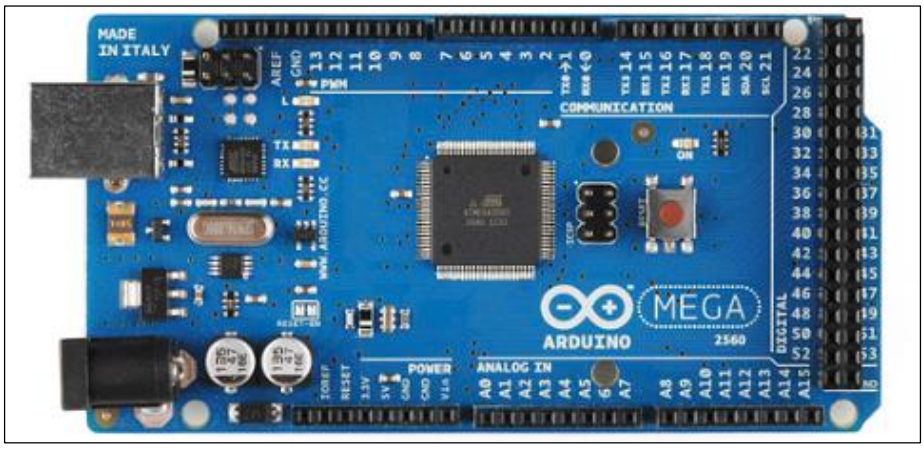


Figura 1.14: Arduino Mega
Fuente: (Arduino.cl, 2018)

El Arduino Mega cuenta con 54 pines digitales que pueden ser configurados como entrada o salidas de datos, de estos se pueden utilizar 15 pines como salidas para la modulación por ancho de pulso (García, 2013).En la tabla 1.1 se muestra las características del Arduino Mega.

Tabla 1.1
Características del Arduino Mega

Parámetro	Descripción
Microcontrolador	ATmega2560
Voltaje de entrada	7-12 Voltios (operativo 5 Voltios)
Pines digitales	54 pines digitales (entada/salida)
Pines PWM	15 pines (salidas - dentro de los pines digitales)
Pines análogas	16 pines (entradas)
Memoria flash	256KB
Corriente pines	40 mA (miliamperios)
Corriente Pin 3.3V	50 mA (miliamperios)
Oscilador	16MHz (megahercio)

Fuente: Elaborado por el autor

1.7 Xbee

Es un dispositivo electrónico que ha sido fabricado por DIGI International y es utilizado para dar soluciones de conexión inalámbrica. En el mercado existen los módulos de la serie 1 y la serie 2. Los cuales están diseñados con la misma distribución de pines, pero presentan una desventaja a la hora de utilizarlos ya que estos no son compatibles entre sí, esto se debe a que trabajan con diferentes protocolos y a que utilizan diferentes chipsets. (Ruiz , 2012)

Los módulos de la serie 1 han sido diseñados en base a un chipset de tipo Freescale, el cual es utilizado en redes de trabajo punto a punto y punto a multipunto. En cambio, los módulos de la serie 2 están diseñados en base a un chipset de tipo Ember y son utilizados para trabajar en redes de tipo malla (mesh). Estos módulos son económicos y fáciles de utilizar, entre sus características destacan su bajo consumo de energía y su largo alcance de transmisión con línea de vista. (Ruiz , 2012)

Este módulo funciona con un voltaje de 2.8 a 3.4 voltios y mediante los pines TXD y RXD realiza la transmisión de datos hacia un microcontrolador o plataforma de desarrollo para lo cual requiere de una tarjeta Xbee explorer. Los pines de conexiones de este dispositivo se muestran en la figura 1.15.

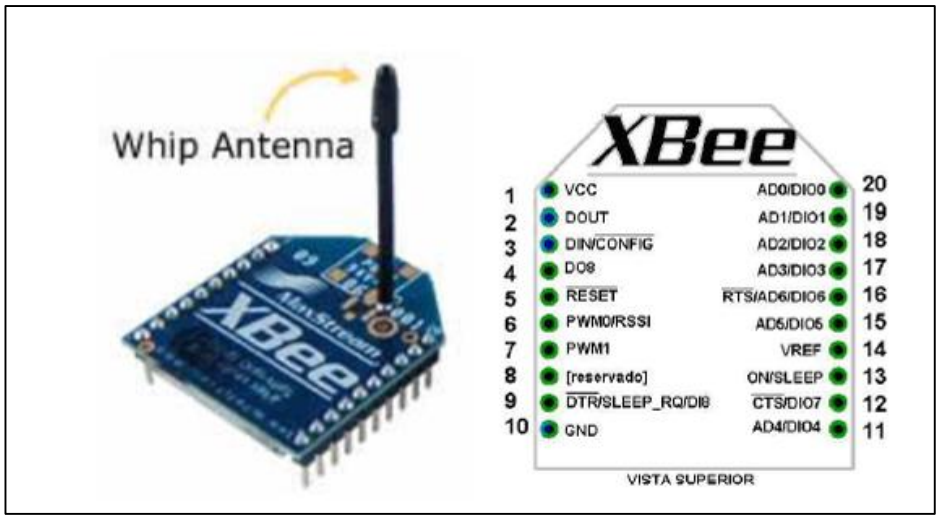


Figura 1.15: Modulo Xbee y sus pines de conexión
Fuente: (Ruiz , 2012)

A continuación, se muestra en la tabla 1.2 las características técnicas del módulo Xbee S1.

Tabla 1.2
Parámetros Xbee

Parámetro	Descripción
Alcance en interiores	30 m (metros)
Alcance con línea de vista exterior	100 m (metros)
Potencia de transmisión	1 mW (milivatio)
Frecuencia	2.4 GHz (gigahercios)
Velocidad de transmisión datos	250 Kbps (kilobits por segundo)
Voltaje de operación	2.8 – 3.4 V (voltios)
Corriente	45 mA (miliamperios)

Fuente: Elaborado por el autor

1.7.1 Zigbee

Es un protocolo de comunicación inalámbrica (reglas para el intercambio de datos) que está basado en el estándar IEEE 802.15.4. Ha sido creado por Zigbee Alliance, Esta tecnología permite la comunicación inalámbrica entre dispositivos electrónicos, sin la necesidad de utilizar algún medio físico entre en transmisor y el receptor. (Oyarce, 2010)

Este protocolo trabaja en una frecuencia en la banda libre de 2.4 GHz (gigahercios). Con respecto al alcance esta depende de la potencia de transmisión y también de la antena que se esté utilizando y además puede ser utilizado para formar una red Zigbee de hasta 65535 equipos. (Oyarce, 2010)

1.7.2 Topología Red Zigbee

La topología básica está conformada por el Coordinador, Routers y dispositivos finales. Estos elementos cumplen una función específica dentro de la red.

- **Dispositivo Coordinador**

Este dispositivo también conocido como nodo, es el encargado de establecer la comunicación y de determinar el área de trabajo personal única (ID PAN) de toda la red. Su función dentro de la red es el enrutado de paquetes. (Oyarce, 2010)

- **Dispositivos Routers**

Es el encargado de crear y mantener los datos electrónicos sobre una red de comunicación inalámbrica y a su vez determinar la mejor ruta de transmisión de datos. Además, es el encargado de retransmitir los paquetes a otros routes y a los dispositivos finales. (Oyarce, 2010)

- **Dispositivos Finales**

Estos dispositivos trabajan atreves de un router o un coordinador, es decir dependen de un nodo de central. Además por su función dentro de las diferentes topologías de red desplegadas en la figura 1.16, no pueden enviar información a otros dispositivos finales. (Oyarce, 2010).

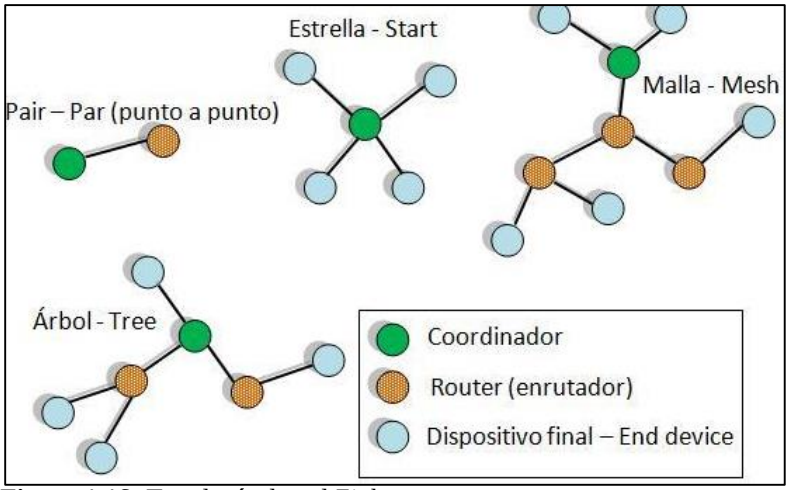


Figura 1.16: Topología de red Zigbee
Fuente: (Plataformas Zigbee, 2014)

Los módulos Xbee son versátiles y pueden ser utilizados en redes punto a punto, punto a multipunto, malla, estrella y mixtas.

1.7.3 Xbee explorer USB

Esta placa electrónica permite utilizar los módulos Xbee tanto de la serie 1 y l la serie 2 de forma directa mediante el puerto USB del computador. Para utilizarlo se debe acoplar el módulo Xbee sobre los pines de placa y también se debe conectar el cable mini USB hacia el computador. (Tdrobotica, 2018)

Esta conexión permite la comunicación serial UART (Transmisor-Receptor Asíncrono Universal) entre el módulo Xbee y una tarjeta Arduino la cual se utiliza para la configuración del enlace inalámbrico, además funciona como interfaz entre un computador y el Xbee. (Tdrobotica, 2018). En la figura 1.17 se muestra el Xbee explorer.

.

Figura 1.17: Xbee explorer USB
Fuente: (Tdrobotica, 2018)

1.7.4 Modo de funcionamiento

El modo de funcionamiento permite conocer la forma en que un microcontrolador se comunica con el módulo Xbee mediante de la interfaz serial.

Al utilizar la interfaz serial se determinar el Modo de funcionamiento para la transmisión de datos. Este es el caso del modo transparente (AT), el cual permite trasmitir datos mediante el pin de salida TX, hacia la dirección de destino previamente configurada y mediante el pin RX recibir cualquier información.

1.8 Servomotor

El servomotor es muy utilizado en aplicaciones como la robótica y el aeromodelismo. En su interior se encuentran mecanismos como el motor de corriente continua y el sistema de control de posición. Además, disponen de tres cables de diferente color: el cable rojo para la alimentación positiva, el cable negro para la conexión a tierra (GND) y el último cable para el control de posición.

El servomotor está compuesto por cinco elementos fundamentales como son: Recubrimiento plástico (a), motor de corriente continua (b), sensor de desplazamiento (c), circuito de control (d), engranajes tipo reductores (e) y el eje final (f). Como se muestra en la figura 1.18.

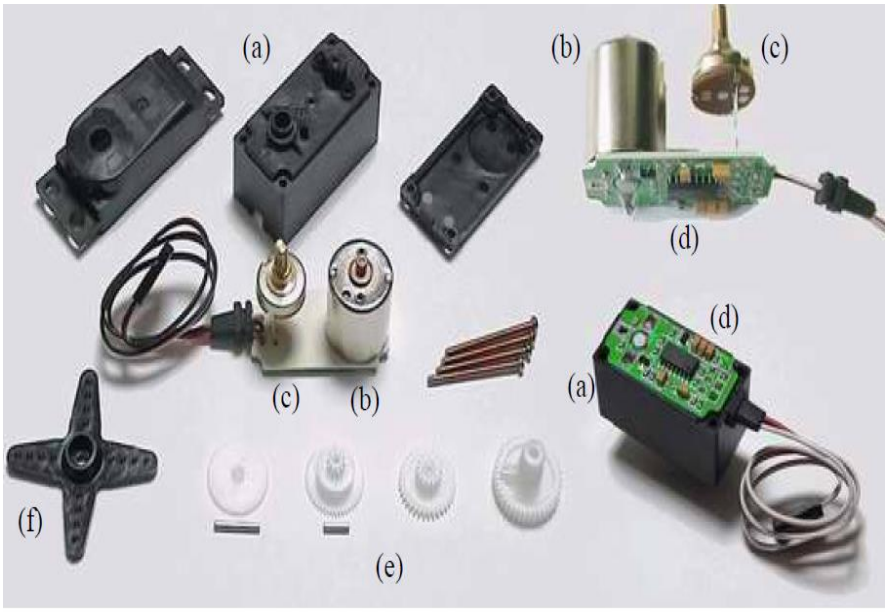


Figura 1.18: Partes del servomotor
Fuente: (Bonell, 2011)

- El Recubrimiento plástico es el material externo el cual por su forma compacta y rígida se encarga de proteger a los componentes internos del servomotor de cualquier golpe o impurezas del ambiente.

- El motor de corriente continua es un mecanismo que brindar movilidad a los engranajes del servomotor. Para su funcionamiento se debe aplicar voltaje de corriente continua en sus dos terminales.

- El sensor de desplazamiento está constituido por un potenciómetro y se encuentra ubicado en el eje de salida del servomotor. Su principal función es detectar la posición angular del motor.
- El circuito de control es una placa electrónica que posiciona al servomotor en la posición deseada mediante el ancho de pulso.

- Los engranajes reductores están encargados de reducir la velocidad de giro del motor. Esto permite desarrollar el torque (momento de fuerza que ejerce el motor sobre el eje de transmisión).

- El eje de salida es un acople plástico, su función es sujetar el servomotor a una superficie fija o móvil. (Bonell, 2011)

1.9 Sensor ultrasónico Ping

Es un dispositivo electrónico que funciona en base a ondas ultrasónicas, por lo cual su espectro audible no puede ser escuchado por seres humanos, esto se produce porque una persona es capaz de escuchar sonidos dentro de un rango de 20 hercios a 20 Megahercios. El sensor ha sido diseñado para medir la distancia del objeto detectado en un rango de 3 centímetros hasta los 3 metros. Además, está constituido por tres terminales; el pin GND como referencia a tierra, el pin de alimentación de 5 voltios y el pin SIG para la entrada y salida de datos. (Herrera, 2009). En la imagen 1.19 se muestra el sensor Ping.

Figura 1.19: Sensor Ping
Fuente: (RobotShop , 2018)

Las características técnicas del sensor de detección se muestran en la tabla 1.3.

Tabla 1.3
Características del sensor Ping

Parámetro	Descripción
Dimensiones	22 x 46 x 16 mm
Rango de medición	3 cm – 3 m
Voltaje de alimentación	5 V
Consumo de corriente	30 - 35 mA
Frecuencia de ultrasonido	40kHz

Fuente: Elaborado por el autor

1.9.1 Funcionamiento del sensor Ping

Este sensor realiza mediciones entre objetos móviles y objetos estáticos. Su funcionamiento depende de los parámetros configurados en un microcontrolador, y para esto se debe establecer en el microcontrolador un pin que trabaje como entrada y salida. Como se muestra en la figura 1.20.

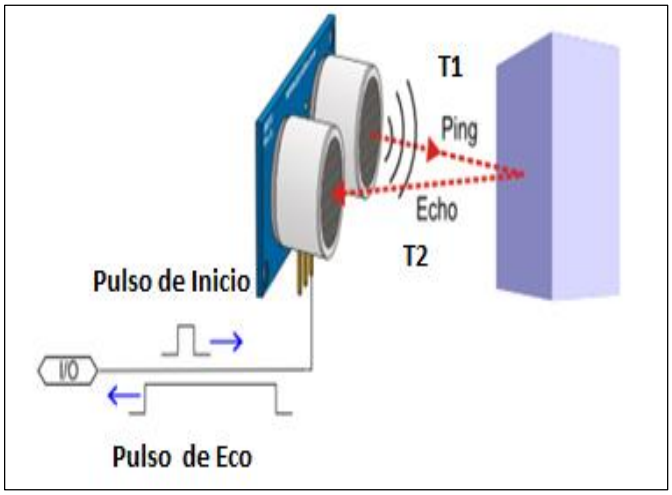


Figura 1.20: Funcionamiento del sensor Ping
Fuente: (Parallax, 2018)

El primer paso consiste en configurar el pin del microcontrolador como salida y enviar una señal negativa, esto genera un nivel lógico bajo en el pin SIG del sensor, después se debe programar un pulso lógico positivo de 5 microsegundos (μs) para activar el funcionamiento del sensor, después que termine este pulso se debe configurar el pin del microcontrolador utilizado como entrada.

Con esta configuración se activa el sensor y transmite durante 200 microsegundos (μs) una ráfaga a 40 kilohercios (kHz) (Herrera, 2009). Esta ráfaga golpea al objeto que tiene al frente generando una señal de rebote, la cual es detectada por el receptor del ultrasónico (Herrera, 2009). En el instante que se envía la ráfaga, el pin de comunicación (SIG) debe estar configurado en estado lógico alto y permanecer en ese estado por un tiempo proporcional al tiempo de vuelo, al chocar con un objeto, la señal resultante ingresa de vuelta por el pin de comunicación (SIG) para proceder a calcular la distancia del objeto detectado. (Herrera, 2009)

1.10 Baterías de LiPo

Estas baterías están hechas de un componente químico conocido como polímero de litio, esto les permite almacenar altas densidades de energía y una mayor vida útil a diferencia de otros componentes químicos. Son muy utilizadas en equipos de electrónica como: Drones, robots, celulares, etc. Dentro de sus ventajas destacan: el suministro prolongado de energía, carga rápida, menor peso y dimensiones reducidas. En la figura 1.21 se muestra la batería de Lipo (Polímero de litio).

Figura 1.21: Batería de LiPo
Fuente: (T-BEM, 2018)

Estas baterías pueden estar conformadas por una o por varias celdas de 3.7 voltios las cuales vienen interconectadas en serie o en paralelo. Si las celdas de energía vienen conectadas en paralelo tendrán mayor capacidad de corriente. La batería de LiPo viene expresada en mAh, (miliamperios hora), es por esto que a mayor capacidad mayor será su tamaño y peso. Además, es recomendable mantener el voltaje de salida en un rango de 3 a 4.2 voltios con el propósito de extender la vida útil de la batería. (T-BEM, 2018).

1.10.1 Convertidor de voltaje LM2596

El circuito LM2596 (figura 1.22), es un convertidor de voltaje de tipo DC-DC (corriente directa a corriente directa). Su función es transformar un nivel de voltaje alto en un voltaje de menor nivel. Este regulador es de tipo conmutado por lo cual representa un alto nivel de eficiencia energética (producir más con menos energía), ya que a diferencia del regulador de tipo lineal como el LM7805 el cual no es eficiente energéticamente. (Naylamp, 2018).

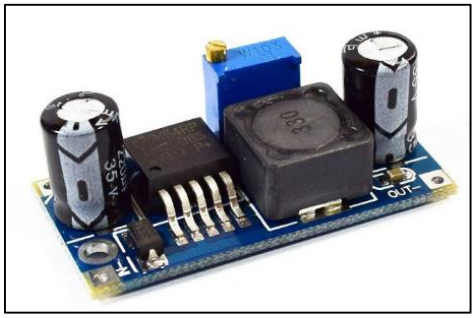


Figura 1.22: Convertidor LM2596
Fuente: (Naylamp, 2018)

Los conmutadores DC-DC se clasifican por su voltaje de salida en: reductores (Step-Down), elevadores (Step-Up) y reductores-elevadores (Step-Up-Down). El convertidor LM2596 es un regulador de tipo conmutado reductor (Step-Down), que presenta una alta eficiencia de conversión y un bajo voltaje de rizado. Este módulo permite obtener un voltaje regulado a partir de una fuente con un voltaje mayor, por ejemplo: obtener 5 voltios a partir de una fuente de 12 voltios. Para un buen funcionamiento el voltaje de entrada debe ser superior al voltaje de salida por lo menos en 1.5 voltios (V) y con esto se evita problemas de eficiencia y rendimiento. (Naylamp, 2018)

A continuación, en la tabla 1.4 las características técnicas del LM2596.

Tabla 1.4
Especificaciones técnicas LM2596

Parámetro	Descripción
Voltaje entrada	4.5 a 40 V
Voltaje salida	1.23 a 37 V
Corriente salida	3 A

Fuente: Elaborado por el autor

1.10.2 Consumo eléctrico

En esta parte se describe las fórmulas para calcular la capacidad de la batería a utilizar (Cavasassi, 2019). La ecuación número 4 se utiliza para determinar la potencia de consumo (C), para esto se debe determinar el valor de la potencia de los componentes electrónicos (Pc) y también de funcionamiento en horas (t).

$$C = Pc * t \qquad \text{Ecuación 4}$$

- C = Consumo [Wh]
- Pc = Potencia del circuito [W]
- t = tiempo (horas)

La ecuación 5 se emplea para establecer la cantidad de la descarga de la batería, este parámetro ha sido definido en 75%. Esto quiere decir que después de agotar el consumo (C), deberá quedar en la batería una capacidad restante del 25% ($P_{D\ (25\%)}$), este parámetro permite extender la vida útil de la batería.

$$P_{D\ (25\%)} = C * (0.25) \qquad \text{Ecuación 5}$$

- $P_{D\ (25\%)}$ = Capacidad restante al 25% [Wh]
- C = consumo [Wh]

La ecuación 6 se utiliza para sumar los valores obtenido en el consumo (C) y la capacidad restante PD (25%).

$$\boldsymbol{P_R = C + P_{D\ (25\%)}} \qquad \text{Ecuación 6}$$

- P_R = Potencia resultante [Wh]
- C = Consumo [Wh]
- $P_{D\ (25\%)}$ = Capacidad restante al 25% [Wh]

La ecuación 7 se utiliza para calcular la potencia de la batería. Para esto se debe dividir la potencia resultante (PR), con el voltaje de la batería a utilizar (V).

$$\boldsymbol{C(Bat) = \frac{P(R)}{V}} \qquad \text{Ecuación 7}$$

- C(Bat) = Capacidad de la batería (Ah)
- P(R) = Potencia resultante (Wh)
- V = Voltaje de la Batería (V)

A continuación, se muestra la fórmula para calcular el tiempo de funcionamiento del circuito eléctrico. (Méndez, 2018)

$$\boldsymbol{Tiempo\ de\ trabajo\ = \frac{C}{I}\ (horas)} \qquad \text{Ecuación 8}$$

- C= Capacidad de la batería expresa en amperios hora
- I = Representa el consumo de corriente del circuito expresado en amperios

1.11 Cargador iMAX B6

El Cargador B6 se utiliza para cargar las baterías de LiPo (polímero de litio), este dispositivo ajusta la corriente de alimentación de forma automática, durante el proceso de carga y descarga. Además, impide la sobrecarga de voltaje ya que al exceder el rango de carga esta podría explotar por negligencia del usuario. Este cargador funciona con una fuente de 11 a 18 voltios de corriente continua y puede cargar baterías de LiPo de hasta 6 celdas. (Hobbyking, 2019). Este cargador junto con sus respectivos componentes se muestra en la figura 1.23.

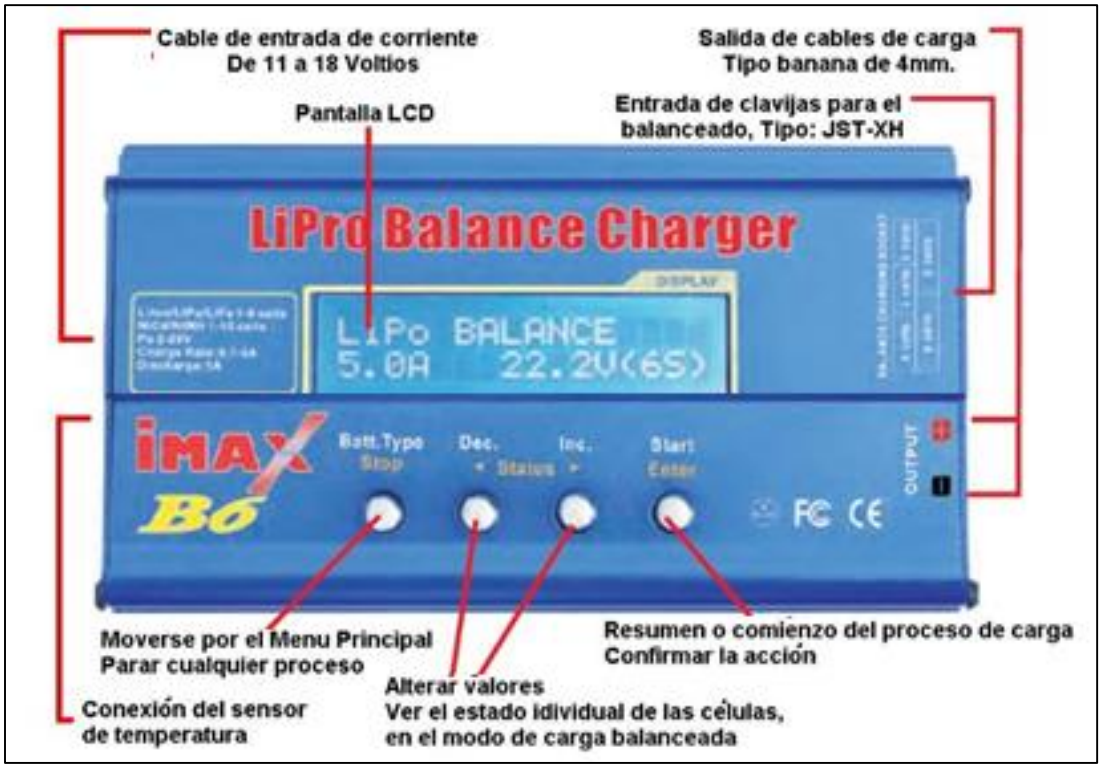


Figura 1.23: Cargador B6
Fuente: (Hobbyking, 2019)

1.12 Software Arduino

El software Arduino también conocido como entorno de desarrollo integrado (IDE), ha sido diseñado para desarrollar proyectos de electrónica desde un nivel básico hasta un nivel avanzado a nivel mundial. Arduino está basado en un sistema de código abierto en donde los usuarios acceden de forma gratuita a los archivos de programación de cualquier proyecto para utilizarlos. (Aprendiendo Arduino, 2017).

En la figura 1.24 se muestra la ventana de inicio del software Arduino.

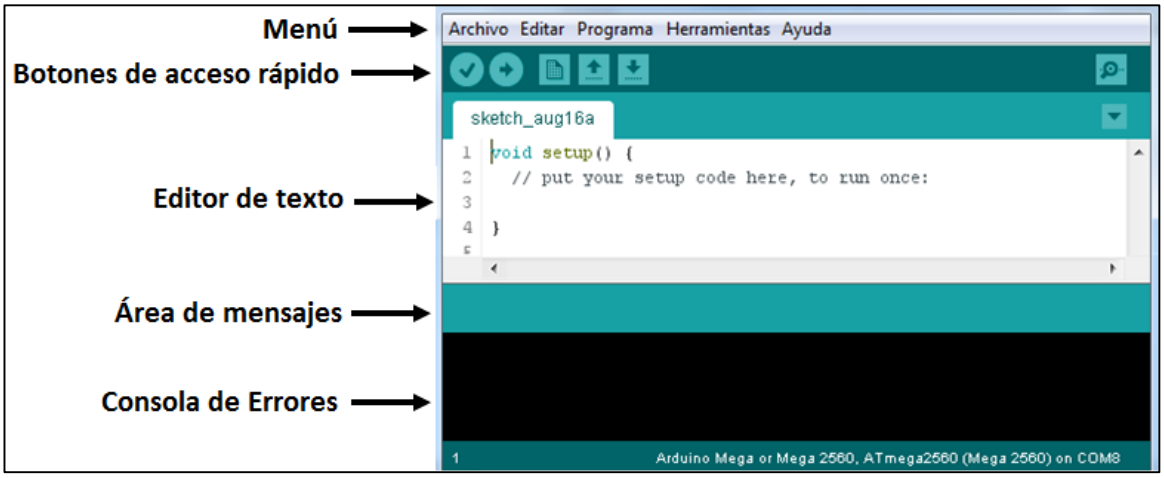


Figura 1.24: Procedo del diseño electrónico
Fuente: (Aprendiendo Arduino, 2017)

El software Arduino contiene herramientas para la programación entre estas: el editor de código en donde se crea el boceto de programación, los botones de acceso rápido para compilar y cargar la programación a la placa Arduino, el área de mensajes para observar comentarios de la plataforma y la consola de errores. También cuenta con una serie de librerías que sirven para conectar cualquier dispositivo electrónico con la tarjeta Arduino de forma sencilla. (Aprendiendo Arduino, 2017)

1.13 Software Proteus

Para el diseño electrónico de este proyecto se utilizará el software Proteus, el cual permite diseñar los diagramas de la parte electrónica y la elaboración del diagrama del circuito impreso. Proteus VSM (Virtual System Modelling), está compuesto por una amplia librería de dispositivos y por un gestor de reglas de diseño que permite: la simulación de microcontroladores, enrutado manual, enrutado automático y visualización de imágenes en tres dimensiones. Entre sus funciones principales están: el diseño electrónico, construcción de circuito impresos y la simulación de los componentes en tiempo real.

El software cuenta con dos módulos: el primero es el módulo Isis en el cual se diseña los esquemas electrónicos y permite realizar la simulación. El otro modulo se lo conoce como Ares, una herramienta que permite el diseño de circuitos impresos. Además, cuenta con la visualización en tres dimensiones (3D) para verificar el trabajo realizado. (Figura 1.25)

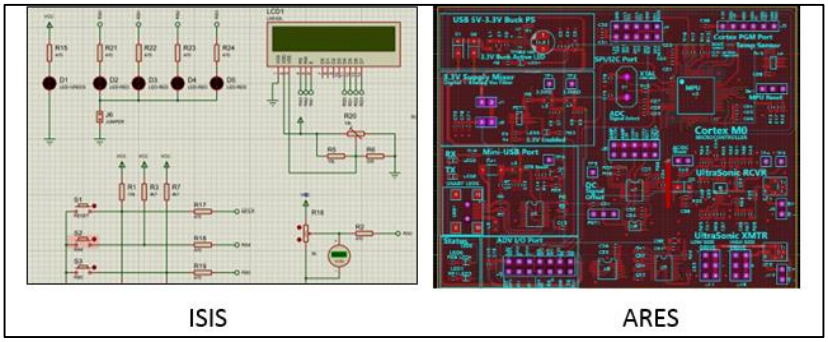


Figura 1.25: Módulos Isis y módulo Ares
Fuente: (Hubor, 2015)

La fabricación de un equipo electrónico se lo puede dividir en cuatro etapas, las cuales permiten al usuario construir una gran variedad de proyectos y en el proceso depurar errores sin riesgos de estropear los dispositivos electrónicos. En la primera etapa se desarrolla el diseño del circuito de acuerdo a las características de funcionamiento requeridas. En la segunda etapa se verifica su correcto funcionamiento mediante la simulación del circuito, aquí es donde se depuran posibles errores. En la tercera etapa se realiza el diseño del circuito impreso y en la última etapa se procede a la construcción física del circuito. (Hubor, 2015)

1.14 Software Visual Studio

El software Visual Studio es una plataforma creativa también conocida como un entorno de desarrollo (IDE). Es compatible con varios lenguajes de programación como Visual Basic., Python, entro otros. (Microsoft , 2019)

Visual Studio cuenta con herramientas digitales para editar, depurar y compilar el algoritmo de programación. Esta plataforma se utiliza para el desarrollo de programas, páginas web y aplicaciones móviles. (Microsoft , 2019).

Entre los lenguajes de programación que utiliza Visual Studio se hace referencia al Visual Basic. Este se utiliza para la creación de aplicaciones y que está orientado a objetos. Este lenguaje de programación está basado en Microsoft .NET Framework (entorno de trabajo para el desarrollo del software) por lo cual se beneficia de la seguridad y el intercambio de información entre idiomas. (Microsoft , 2019)

1.15 Software XCTU

El software XCTU (*XBee Configuration and Test Utility*), es una plataforma gratuita diseñada para administrar módulos DIGI de radio frecuencia (RF), es compatible con Windows, MacOS y Linux. Esta interfaz cuenta con herramientas que facilitan la configuración y prueba de los módulos Xbee mediante un enlace en serie. (DIGI, 2019). Los módulos Xbee se utilizan para transmitir y recibir señales de radio frecuencia. La empresa DIGI produce una amplia variedad de estos módulos los cuales pueden ser utilizados dependiendo de los requisitos para el enlace inalámbrico. (DIGI, 2019).

1.16 Software AutoCAD

Este software es de tipo CAD (*diseño asistido por computadora*) y es utilizado por arquitectos, ingenieros y por estudiantes para crear dibujos en dos dimensiones (2D) y tres dimensiones (3D). Es compatible con los sistemas operativos Windows y MacOS. Esta plataforma de diseño permite crea, edita modelos sólidos y añadir anotaciones de texto. (Autodesk, 2019).

1.17 Software Solidworks

Solidworks es un software que permite el diseño asistido por computadora (*CAD*), para el modelado mecánico en dos dimensiones (*2D*) y en tres dimensiones (*3D*). Este programa es compatible con Windows y es utilizado para modelar piezas y conjuntos, además permite extraer los planos técnicos en las perspectivas que se requiera, por ejemplo: vista frontal, vista lateral y superior.

CAPÍTULO 2

MARCO DETODOLOGICO

En este capítulo se describen los aspectos metodológicos y prácticos que se utilizan para la Implementación de un robot bípedo con tecnología Xbee. En esta parte se describe el tipo de investigación utilizada, la metodología seleccionada y las técnicas e instrumentos utilizados.

2.1 Tipo de investigación

En el presente proyecto se emplea la investigación aplicada, la cual se enfoca en la resolución de problemas dentro de la sociedad o del sector productivo. Su uso consiste en utilizar el conocimiento teórico que ha sido adquirido durante la formación académica para la elaboración de un producto que responda a las necesidades encontradas. (Lozada, 2014).

El tema de este proyecto consiste en la implementación un prototipo de robot bípedo con tecnología Xbee para la detección de obstáculos predeterminados. Este tema ha sido desarrollado para dar a conocer un instructivo sobre la elaboración de un robot bípedo, ya que existen personas que en la elaboración de este tipo de robot presentan diversos vacíos tanto en la parte teórica como en la parte práctica.

2.2 Técnicas e instrumentos de recolección de datos

En el primer capítulo se utiliza el método de la revisión documental, el cual se utiliza para desarrollar el contenido de la fundamentación teórica de este proyecto. Según Ortega Daniel, este método se utiliza para la revisión y registro de documentos referente al tema de investigación. (Ortega, 2012)

El método de la revisión documental ha sido complementado consecuentemente con la técnica llamada revisión bibliográfica. Mediante esta se ha obtenido información relevante al tema de investigación, por ejemplo: la robótica móvil, la locomoción bípeda, Arduino, servomotor, sensor ultrasónico, la batería de litio, entre otros temas.

En el segundo capítulo se utiliza la técnica de la observación documental. Esta técnica es aplicada para seleccionar una metodología que esté acorde al proyecto de investigación. Para lo cual se utiliza documentos y publicaciones web con información referente a las fases que conforman la metodología predeterminada. Según Balestrini Miriam esta técnica permite desarrollar un análisis profundo de las fuentes de información. (Balestrini, 2006).

La técnica de la observación documental es complementada con el instrumento llamado registro de información. Según Arias Fidias el instrumento es cualquier recurso utilizado para guardar la información sobre el proyecto de investigación. (Arias, 2006)

En el tercer capítulo se utiliza el método de la modelación. Según Díaz Vladimir la modelación es un proceso para la creación o representación del objeto a estudiar. (Díaz , 2013). Este método es aplicado para relacionar las especificaciones y objetivos propuestos en este proyecto con el modelo del prototipo a diseñar. Este método será aplicado en la fase de diseño electrónico y en la parte del diseño de la estructura.

En el cuarto capítulo se emplea el método experimental, el cual permite la reproducción real del robot utilizando los componentes mecánicos y el software adecuado. Este método permite verificar el funcionamiento del prototipo y a su vez corregir las falencias encontradas.

2.3 Metodología seleccionada

En el presente proyecto se emplea la metodología planteada por Angulo, J. (1986) debido a la concordancia de las fases del autor con los objetivos del proyecto. (Chacin & Roseli, 1999). Esta metodología se refiere a 7 etapas las cuales son: 1. Definiciones las especificaciones; 2. Esquema general del hardware; 3. Diagrama general del software; 4. Implementación del hardware; 5. Programación y configuración; 6. Integración del hardware con el software; 7. Construcción del prototipo y pruebas finales.

A continuación, se describe la aplicación de cada una de las fases:

1) Especificaciones generales

En esta fase se describe las características del sistema a desarrollar y se realiza el esquema para la representación de la propuesta.

2) Esquema General del Hardware

En esta fase se realiza el esquema del hardware mediante un diagrama de bloques para representar las etapas del sistema y determinar su funcionamiento.

3) Diagrama General del Software

En esta parte se desarrolla el diagrama de flujo para indicar el funcionamiento del algoritmo de forma general. En esta parte se desarrolla el software a utilizar por el del sistema electrónico.

4) Implementación del Hardware

En esta fase se elabora el circuito de control, para esto se debe verificar las características técnicas de los componentes a utilizar, y de la misma forma las especificaciones de los fabricantes

5) Programación y configuración

En esta fase se realiza el algoritmo de programación y se configura los parámetros de funcionamiento del software a emplear.

6) Integración del Hardware con el Software

Consecutivamente se procede a realizar las pruebas de conexión entre el hardware y el software para comprobar el enlace.

7) Construcción del Prototipo y pruebas finales

En esta fase, se obtiene el dispositivo final con los respectivos circuitos electrónicos y los programas para llevar a cabo el funcionamiento general del sistema.

CAPÍTULO 3

PROPUESTA

En este capítulo se describe el proceso para el diseño del robot bípedo, en donde se detalla la parte mecánica, la parte electrónica y la programación del robot. Además, se describe el uso de las herramientas utilizadas tanto de software y hardware.

3.1 Especificaciones generales

Este proyecto permite conocer el proceso para la elaboración de un prototipo de robot bípedo con tecnología Xbee el cual será capaz de desplazarse sobre una superficie plana evadiendo obstáculos predeterminados.

El prototipo constará de un sensor de proximidad de tipo ultrasónico que estará ubicado en la cabeza del robot. Su función es detectar objetos dentro del rango determinado, para esto el sensor emite un sonido y mide el tiempo que tarda la señal en regresar. Esta información es transmitida hacia la tarjeta Arduino y analizada por el software para enviar las ordenes correspondiente hacia el robot.

En la parte referente al software, se emplea la plataforma Arduino, el cual es un lenguaje de programación alto nivel y es empleado para el desarrollo del algoritmo de control del robot. De igual forma se emplea una interfaz de usuario creada mediante el software Visual Studio para emitir las ordenes de movimiento de forma inalámbrica mediante la tecnología Xbee. Con esta interfaz inalámbrica el usuario será capaz de dirigir el movimiento del robot en tres direcciones (adelante, izquierda y derecha).

Este proyecto esta mayormente enfocado hacia las personas que desconocen el proceso de elaboración un robot bípedo además está orientado a mostrar el funcionamiento de la parte electrónica con respecto a la tarjeta Arduino y al sensor de proximidad.

En la figura 3.1 se muestra el esquema del proyecto a desarrollar. Este proyecto está conformado por un computador con una interfaz de usuario que utiliza un Xbee "A" para establecer la comunicación inalámbrica con el Xbee "B" ubicado en el robot.

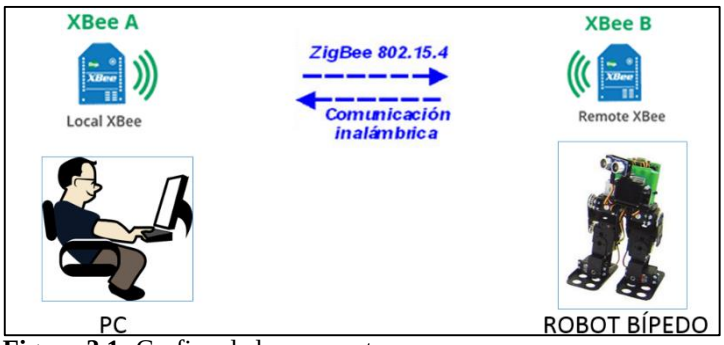


Figura 3.1: Grafico de la propuesta
Fuente: Elaborado por el autor

A continuación, se detalla las especificaciones tomadas en cuenta para el proceso de elaboración del prototipo.

3.2. Parámetros para la estructura del robot

El diseño del robot está dividido en tres partes, las cuales están referenciadas en el parámetro de movilidad para esto se ha tomado en cuenta: peso tamaño y tipo de material.

- **Modelo del robot**

En la elaboración de este proyecto es indispensable determinar la forma del robot bípedo. Por lo cual se ha establecido ciertos parámetros como son: tamaño, peso y grados de libertad. Estos aspectos son necesarios para optar por el modelo más adecuado. En este caso se ha escogido el modelo BRAT de Lynxmotion, ya que esta estructura presenta un parámetro de estabilidad adecuado para el desarrollo de este proyecto.

- **Material para la estructura**

Para determinar el material óptimo necesario para la elaboración del robot se ha realizado un análisis sobre los posibles materiales a utilizar. Entre estos se encuentran los siguientes: Acrílico, Aluminio, Tol galvanizado. En la tabla 3.1 se muestra la comparación de estos materiales.

Tabla 3.1
Comparación de materiales

Características/material	Acrílico	Aluminio	Tol galvanizado
Espesor (milímetros)	3mm	1mm	1mm
Precio (dólares)	$ 60	$ 20	$ 10
Dimensiones (metros)	1.22m x 2.44m	1m x 1m	1m x 1m
Forma de doblar	Calor	Manualmente	Manualmente
Tipo de material	Frágil	Maleable	Maleable
Limite Fluencia	442.23 kg/cm^2	1937.46 kg/cm^2	2319 kg/cm^2

Fuente: Elaborado por el autor

Mediante las características de los materiales expuestos en la tabla 3.1 se ha determinado los siguientes resultados:

a) El acrílico es un material frágil y costoso
b) El aluminio es un material maleable y costoso
c) El tol galvanizado es un material maleable y económico

Al realizar este análisis se ha determinado que el material más adecuado para la elaboración del robot bípedo es el tol galvanizado. Este material será doblado y cortado para formar el robot. Entre estas piezas están las articulaciones para representar el movimiento mediante un servomotor y los eslabones para formar la base que sostiene a una articulación.

- **Calculo de grados de libertad**

Para determinar el número de grados de libertad (GDL) del robot se debe se utiliza la ecuación de Grübler. Los parámetros de esta ecuación están especificados en el capítulo 1 en el tema 1.4.1 elementos fundamentales.

El primer paso es determinar el número de eslabones como se muestra en la figura 3.2 en la izquierda. El segundo paso es determinar el número de uniones de 1 GDL como se muestra en la figura 3.2 en la derecha.

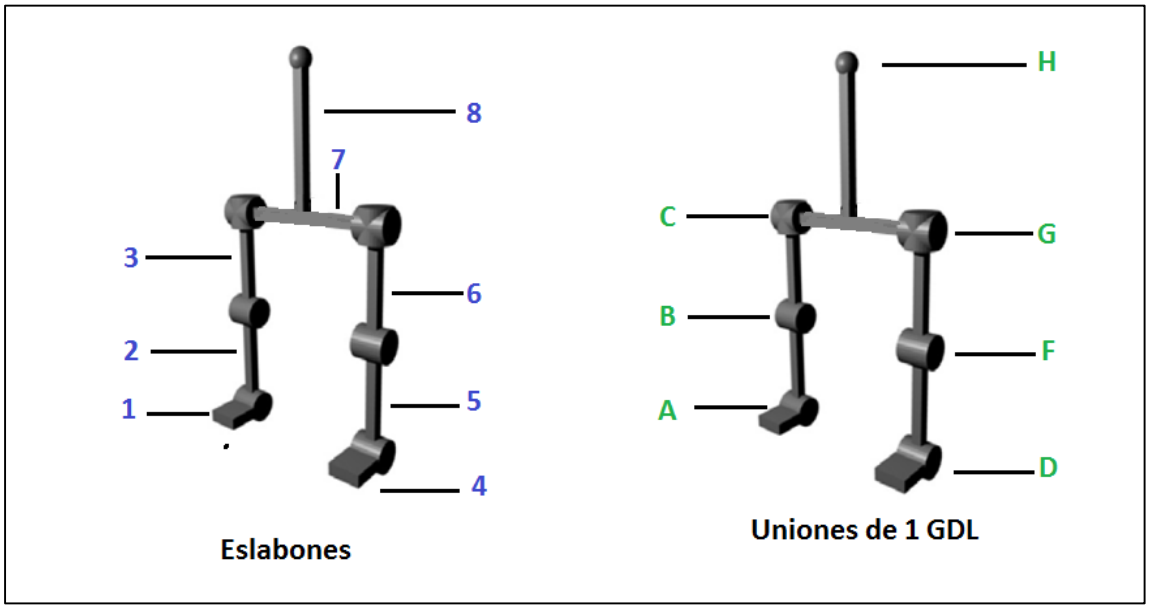


Figura 3.2: Numero de eslabones y GDL
Fuente: Elaborado por el autor

$$GDL = 3\,(n - 1) - 2j1 - j2 \qquad \text{Ecuación 1}$$

$$GDL = 3\,(8 - 1) - 2(7) - 0$$

$$GDL = 3\,(7) - 14$$

$$GDL = 7 \text{ grados de libertad}$$

Al realizar los cálculos respectivos se determinó mediante la ecuación de Grübler que el prototipo debe ser formado por 7 grados de libertad.

- **Diseño de piezas**

Las piezas del robot han sido diseñadas mediante el software AutoCAD, con el propósito de mostrar la forma preliminar de las diferentes piezas entre estas: cadera, rodillas y tobillos. Para esto se ha tomado como modelo de ejemplo el robot BRAT (Robot Articulado de Trasporte Bípedo), mediante este robot se muestra la ubicación de las piezas a elaborar.

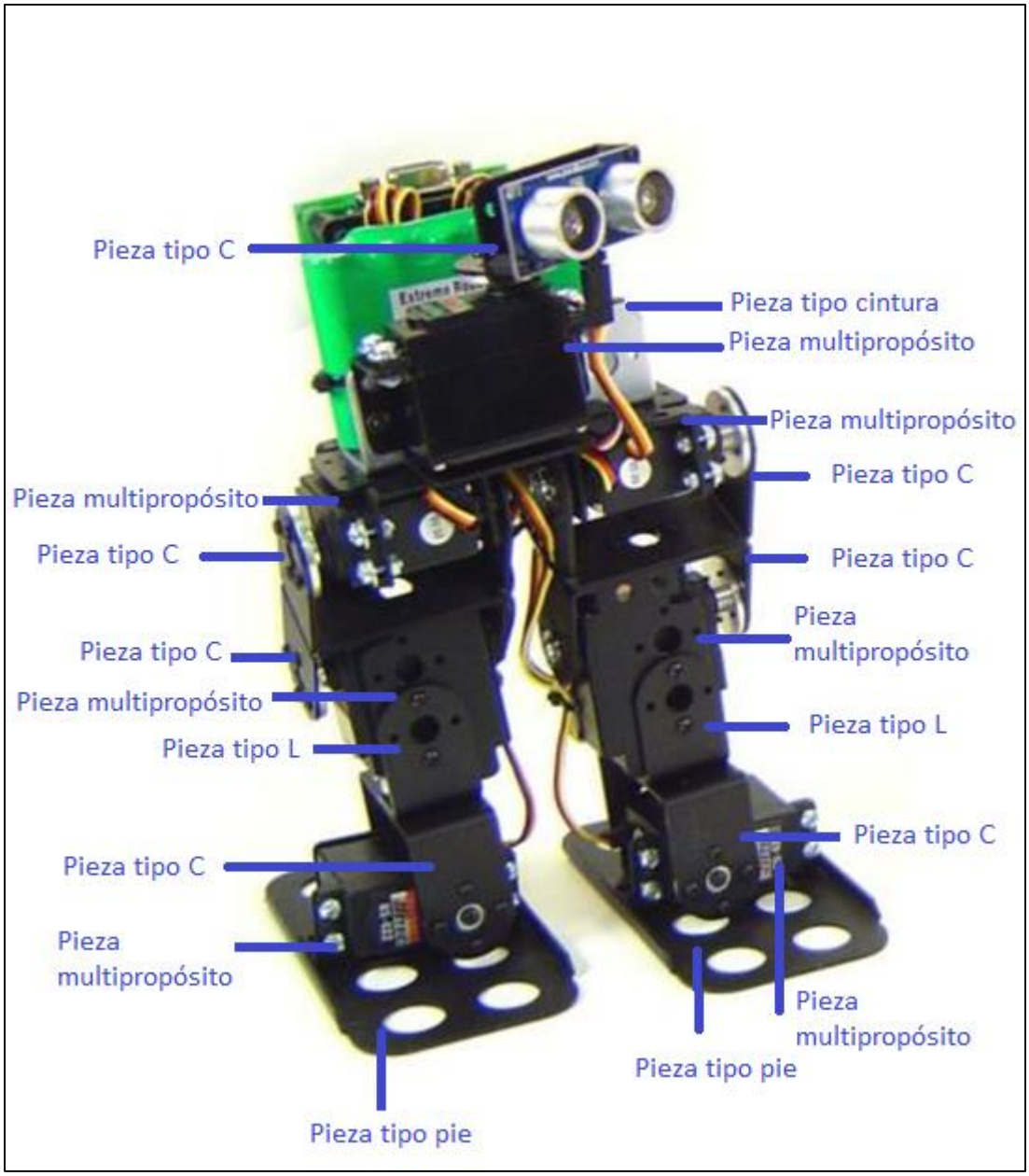


Figura 3.3: Robot BRAT
Fuente: Elaborado por el autor

A continuación, se muestra en la figura 3.4 la pieza multipropósito, su función es sujetar el servomotor y sirve como base para una articulación. Se debe realizar 7 piezas de similar forma porque el robot a elaborar cuenta con 7 grados de libertad, por lo tanto, a cada servomotor le corresponde una pieza multipropósito.

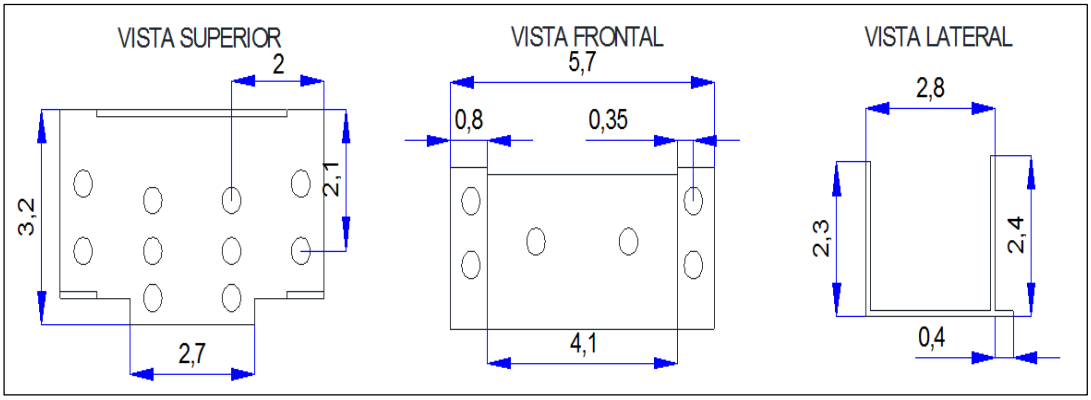


Figura 3.4: Pieza multipropósito
Fuente: Elaborado por el autor

En la figura 3.5, se muestra la pieza tipo *"C"*, su función es sujetar la parte móvil del servomotor y formar una articulación. Para armar el robot se requiere de 7 piezas de similar forma. Se debe realizar 7 piezas de similar forma porque el robot a elaborar cuenta con 7 grados de libertad, por lo tanto, a cada servomotor le corresponde una pieza tipo "*C"*.

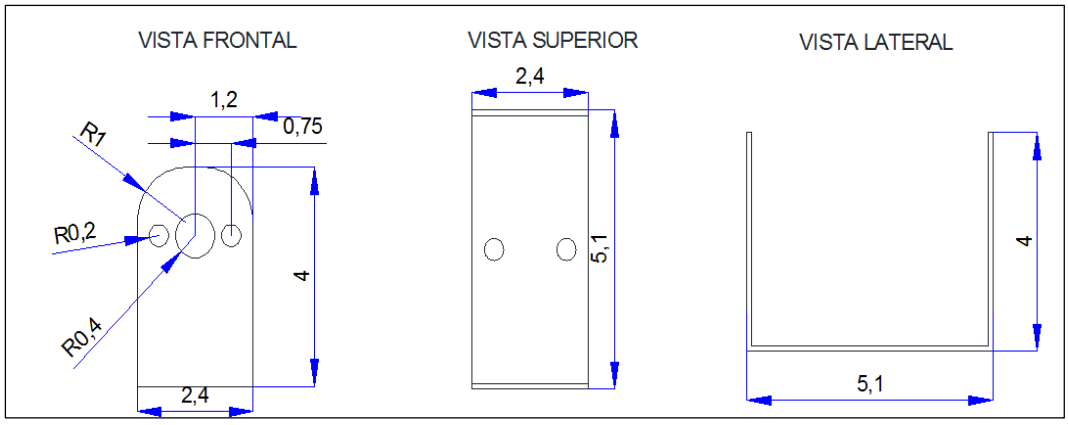


Figura 3.5: Pieza tipo *"C"*
Fuente: Elaborado por el autor

En la figura 3.6, se muestra la pieza tipo *"L"*, la cual se utiliza para sujetar las piezas multipropósito y tipo *"C"*. Para la estructura del robot se requiere dos piezas de similar forma porque se utiliza una por cada pierna.

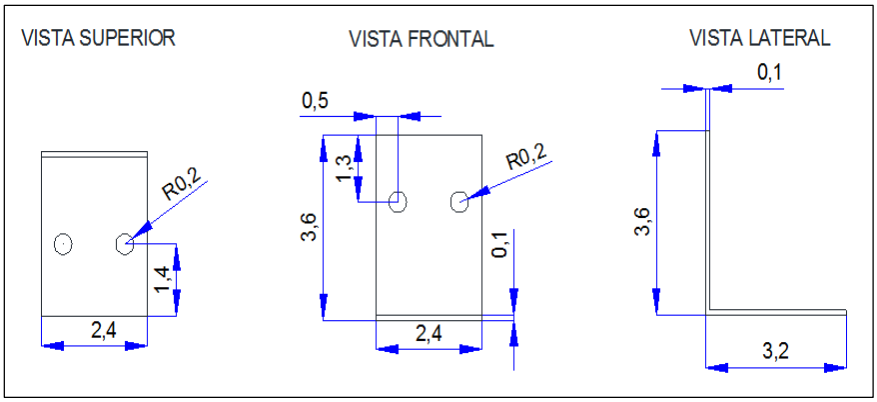


Figura 3.6: Pieza tipo *"L"*
Fuente: Elaborado por el autor

En la figura 3.7, se muestra la pieza tipo pie la cual se utiliza para dar estabilidad al robot y para replicar el desplazamiento del ser humano. Para la estructura del robot se requiere de 2 piezas de similar forma porque el robot a elaborar estará diseñado con 2 pies.

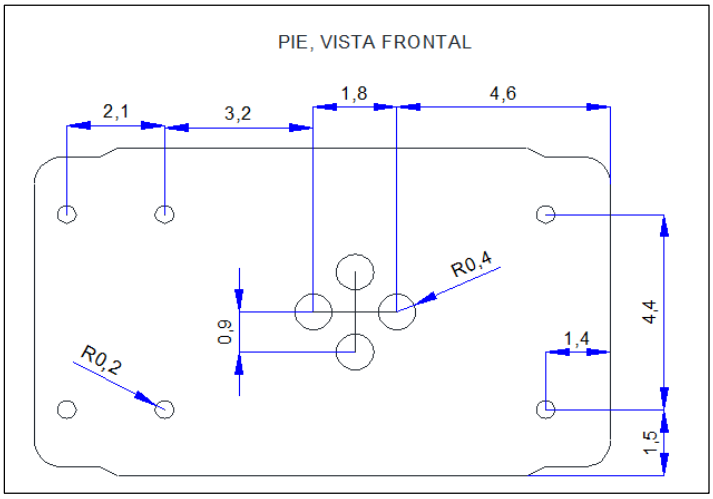


Figura 3.7: Pieza tipo pie
Fuente: Elaborado por el autor

En la figura 3.8 se muestra la pieza tipo cadera, su función es sujetar las piernas del robot, la batería de alimentación y la placa electrónica. Para esta función se requiere de una pieza de este tipo.

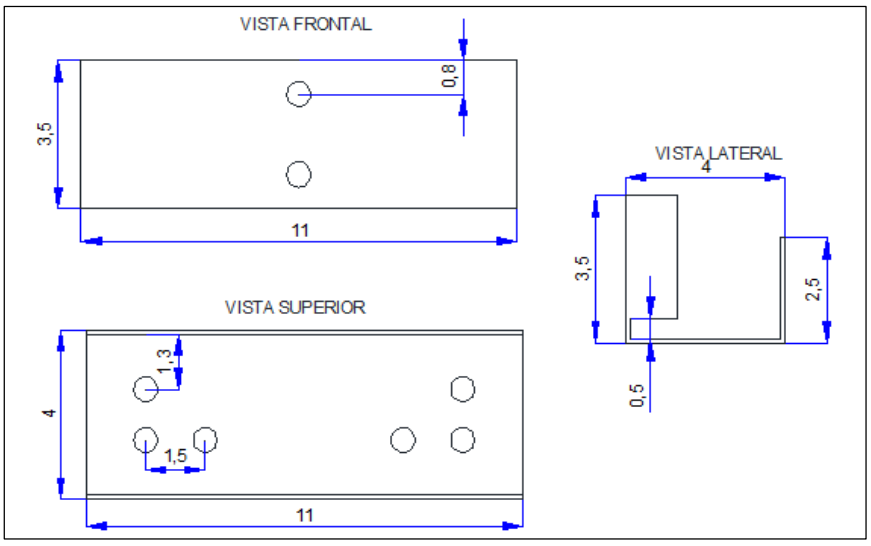


Figura 3.8: Pieza tipo cadera
Fuente: Elaborado por el autor

En la figura 3.9, se muestra la pieza tipo carcaza, la cual se utiliza para sujetar la tarjeta de control del robot bípedo. Para la estructura del robot se requiere de una sola pieza.

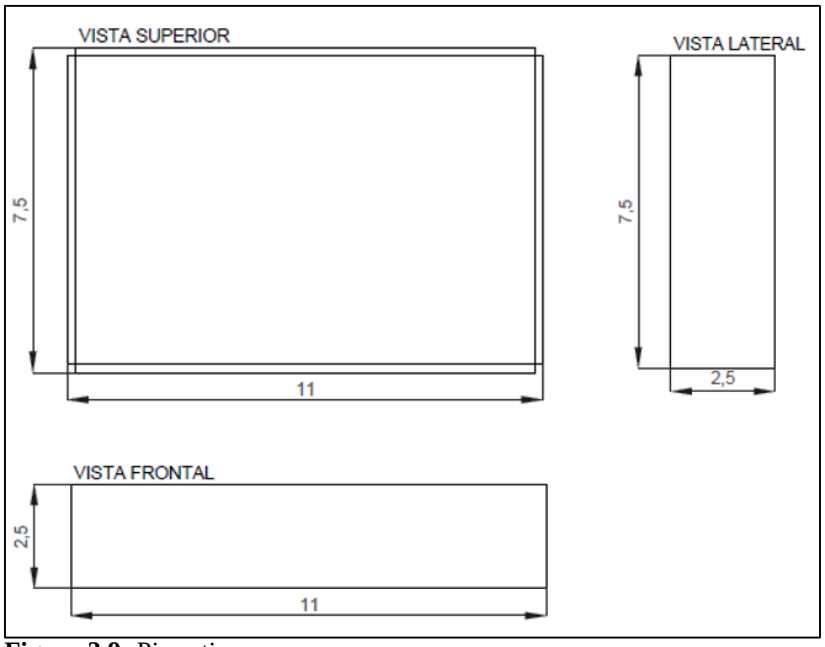


Figura 3.9: Pieza tipo carcaza
Fuente: Elaborado por el autor

3.3 Módulos del sistema electrónico

El esquema general de este sistema está constituido por siete módulos, cada uno con una función específica. De esta forma se ha establecido un orden para dar a conocer de forma adecuada el funcionamiento del sistema electrónico. En la figura 3.10 se muestra el esquema para el proyecto.

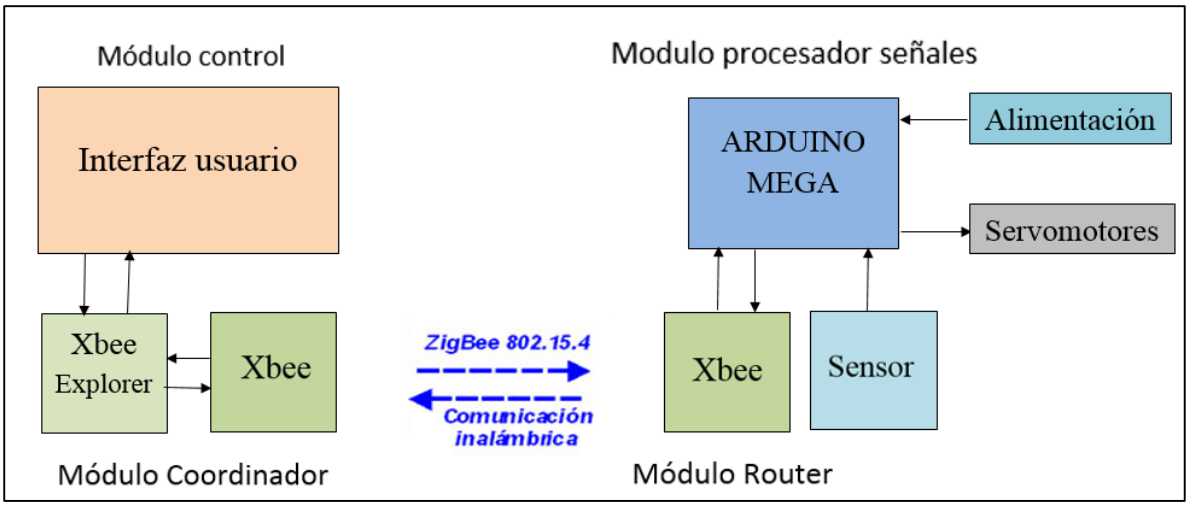


Figura 3.10: Módulos Electrónicos
Fuente: Elaborado por el autor

3.3.1 Módulo de control

El módulo de control está formado por una computadora, cuya función es ejecutar la interfaz de usuario para el control de los movimientos del robot. Este software es desarrollado en la plataforma de Visual Studio mediante el lenguaje de programación Basic.

3.3.2 Módulo coordinador

Es el encargado de enrutar la información desde la interfaz de usuario hacia Xbee router que está ubicado en el circuito de control del robot bípedo. Para el enlace de esta comunicación se utiliza la tarjeta electrónica Xbee Explorer USB, el cual sirve de acople para el módulo Xbee y a su vez sirve de interfaz con el computador.

- **Selección módulo Xbee**

Para la selección del módulo Xbee más adecuado se tiene que tomar en cuenta los parámetros siguientes: Distancia para la comunicación inalámbrica y el lugar de trabajo.

En este caso para determinar la distancia se ha tomado en cuenta la supervisión continuamente del funcionamiento del robot, por lo cual no se requiere de un enlace de largo alcance. En cambio, para determinar el lugar de trabajo se ha tomado en cuenta el área de aplicación del robot, en este caso el robot será utilizado en un aula de clases. En la tabla 3.2 se muestra tres tipos de módulos Xbee.

Tabla 3.2
Comparación Módulos Xbee

Hardware	Nombre	Velocidad de transmisión	Alcance interiores	Lugar de Aplicación
	S1	2.4 GHz	30 m	Interior
	S1 PRO	2.4 GHz	90 m	Interior
	S5	868 MHz	550 m	Exterior

Fuente: Elaborado por el autor

Mediante las características de la tabla 3.2 se ha determinado que el módulo S1 es el más adecuado por qué el alcance inalámbrico de 30 m es suficiente para supervisar continuamente el funcionamiento del robot y porque este módulo ha sido diseñado para ser utilizado lugares cerrados lo cual es adecuado ya que este proyecto ha sido elaborado para explicar el funcionamiento de un robot bípedo en salones de clases, laboratorios o centros de exposición.

3.3.3 Módulo router

El módulo router está formado por un Xbee S1, su función es mantener la comunicación inalámbrica con el modulo coordinador. El enlace entre los módulos Xbee es de tipo hall dúplex, es decir, permite enviar información de forma bidireccional pero no de forma simultánea. Se utiliza la conexión hall dúplex para enviar los datos obtenidos desde el sensor hacia la interfaz de usuario y para enviar los comandos de movimiento desde la interfaz de usuario hacia el robot.

3.3.4 Módulo procesador de datos

Este módulo está formado por un Arduino y su función es controlar el movimiento de los actuadores mecánicos, procesa los datos enviados desde el sensor ultrasónico y recibe la alimentación suministrada desde una batería. Esta tarjeta utiliza el módulo router para recibir la información enviada desde el módulo coordinador.

- **Selección módulo Arduino**

Para la selección de la tarjeta Arduino se ha tomado en cuenta que este dispositivo debe poseer las siguientes características técnicas: puerto de comunicación USB, espacio de memoria adecuado y pines de conexión para el acople de una placa electrónica. En la tabla 3.3 se muestra las características de tres módulos Arduino.

Tabla 3.3
Comparación Módulos Arduino

Hardware	Nombre	Puerto físico	Memoria Flash	Compatibilidad
	Arduino Nano V3	Mini-USB	32Kb	Protoboards
	Arduino Uno R3	Conexión USB	32Kb	Montaje de placas
	Arduino Mega R3	Conexión USB	256Kb	Montaje de placas

Fuente: Elaborado por el autor

a) Se requiere de un espacio de memoria alto porque el algoritmo de movimientos del robot es extenso.

b) Se requiere un puerto USB de programación, ya que en el desarrollo de este proyecto se debe verificar de forma continua desarrollo del algoritmo para el control del robot.

c) Se requiere pines de conexión para el acople de una placa electrónica, esto se debe a que se requiere una placa adicional para conectar los servomotores, el sensor y el módulo Xbee.

Con estos parámetros se ha determinado que el robot requiere de una tarjeta Arduino Mega la cual es la más adecuada para este proyecto.

3.3.5 Módulo de detección

Su función consiste en detectar objetos predeterminados, Para esto debe configurar el rango de detección mediante el algoritmo de programación.

- **Selección del Sensor**

Para determinar el sensor más adecuado se analizar los diferentes sensores que se muestra en la tabla 3.4.

Tabla 3.4
Comparación Sensor de detección

Hardware	Nombre	Tipo	Distancia	Voltaje
	Sharp	Infrarrojo	20 - 150cm	5V
	Pir	Infrarrojo	3 a 7m	5V
	Ping	Ultrasónico	3cm a 3m	5V

Fuente: Elaborado por el autor

Para la selección del sensor se ha tomado en cuenta las siguientes características: Rango de detección y variación de la iluminación. Se requiere de un corto rango de detección porque el robot funcionara en lugares cerrados (aulas y laboratorios). En este caso los sensores mencionados en la tabla 3.4 cumplen con este parámetro. En cambio, el otro parámetro hace referencia a la variación de luz ambiental que puede darse por flash que emiten las cámaras. En este caso los sensores de tipo infrarrojo (Sharp - Pir) pueden verse afectados ya que estos funcionan mediante rayos infrarrojos en donde el material piroeléctrico del sensor puede activarse. Por lo cual se ha determinado que el sensor ultrasónico (Ping) es el más adecuado para este proyecto ya que su funcionamiento no es afectado por la variación de la iluminación.

3.3.6 Módulo de Actuadores mecánicos

En cuanto a los actuadores se utiliza siete servomotores (figura 3.11), los cuales permiten el movimiento de las articulaciones. Estos componentes están referenciados en el parámetro de movilidad para esto se ha tomado en cuenta el nivel de fuerza a ejercer en cada articulación.

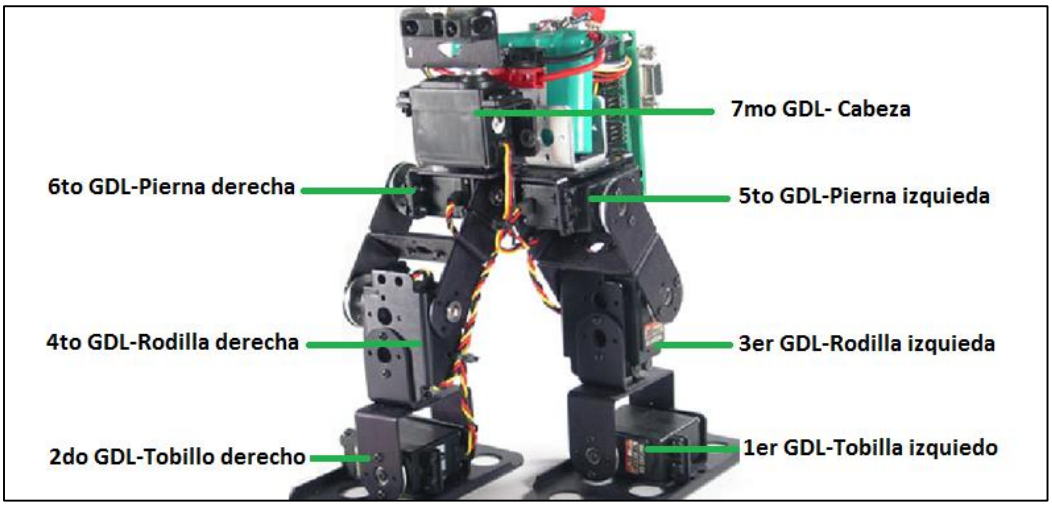


Figura 3.11: Diagrama grafico de la propuesta
Fuente: Elaborado por el autor

- **Cálculos para el torque del servomotor**

Este valor se encuentra al alinear las partes de las articulaciones, como se muestra en la figura 3.12.

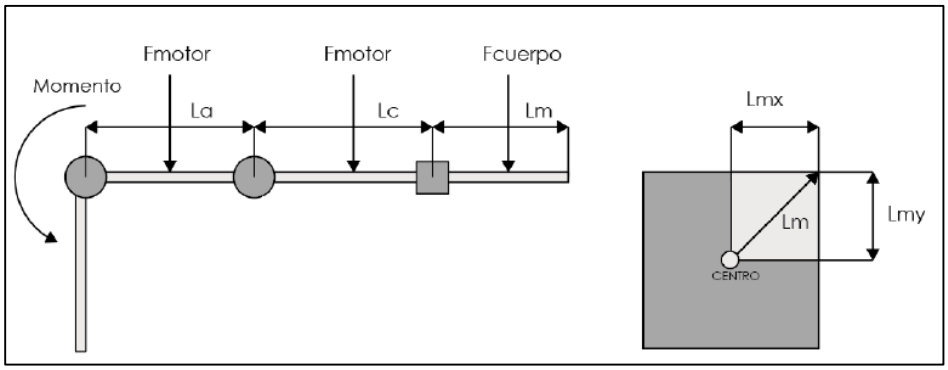


Figura 3.12: Diagrama de la extremidad inferior
Fuente: (Cevallos, 2019)

Los parámetros de la ecuación se especifican en el capítulo 1 en el tema 1.4.1 elementos fundamentales. A continuación, se muestra los datos para determinar el torque en las piernas.

La = 70 mm (Medida del tobillo de servo a servo)
Lc = 60 mm (Medida de la rodilla de servo a servo)
Lmx = 70 mm (Mitad de la estructura central en coordenada "x")
Lmy = 45 mm (Mitad de la estructura central en coordenada "y")
Fmotor = 60 g (Peso servomotor)
Fcuerpo = 1163g (Peso del robot con batería)

$$\boldsymbol{Lm=\sqrt{Lmx^2 + Lmy^2}} \qquad \text{Ecuación 3}$$

$$Lm = \sqrt{(70)^2 + (45)^2}$$

$$Lm = 83{,}21mm$$

$$\boldsymbol{Momento = Fmotor * \frac{La}{2} + Fmotor * (La + \frac{Lc}{2}) + \frac{Fcuerpo}{4} * (La + Lc * Lm)} \qquad \text{Ecuación 2}$$

$$Momento = Fmotor * \left(\frac{3La}{2} + \frac{Lc}{2}\right) + \left(\frac{Fcuerpo}{4}\right) * (La + Lc + Lm)$$

$$Momento = 60 * \left(\frac{3(70)}{2} + \frac{60}{2}\right) + \left(\frac{1163}{4}\right) * (70 + 60 + 83{,}21)$$

$$Momento = 70090{,}80gmm$$

$$Momento = 7{,}0kg.cm$$

El valor obtenido es de 7 kg.cm, por lo tanto, se debe utilizar servomotores de mayor o igual torque tanto en tobillos, rodillas y piernas.

- **Servomotor Turnigy**

Este servomotor se utiliza como articulación para el tobillo y ha sido escogido por su torque de 15.5 kg-cm, ya que su función es soportar todo el peso del robot. (tabla 3.5).

Tabla 3.5
Características Servomotor Turnigy

Parámetro	Descripción
Voltaje de operación	4.8 a 6.0 V
Torque (4.8V - 6V)	15.5 – 17 kg-cm
Dimensiones	40.7 x 20.5 x 39.5 mm
Peso	60.0 gramos

Fuente: Elaborado por el autor

- **Servomotor Power HD**

Este servomotor se utiliza en la rodilla y en la pierna. Ha sido escogido por su torque de 8.6 kg-cm, el cual deberá soportar un porcentaje menos de peso total del robot. (tabla 3.6)

Tabla 3.6
Características servomotor HD

Parámetro	Descripción
Voltaje de operación	4.8 a 6.0 Voltios
Torque	8.6 kg./cm
Dimensiones	41.9 x 20.6 x 41.9 mm
Peso	56.0 gramos

Fuente: Elaborado por el autor

- **Servomotor Hitec**

Este servomotor se utiliza como articulación para la cabeza y ha sido escogido por su torque de nivel bajo ya que su función no se requiere de mayor fuerza. (tabla 3.7)

Tabla 3.7
Características servomotor Hitec

Parámetro	Descripción
Voltaje de operación	4.8 a 6.0 VDC
Torque	3.53 kg./cm
Dimensiones	39.9 x 19.8 x 36.6 mm
Peso	43.0 g

Fuente: Elaborado por el autor

3.3.7 Módulo de alimentación

En la alimentación se debe tomar en cuenta los siete servomotores a utilizar por lo cual se requiere de una batería de alta capacidad, a continuación, se detalla la batería a utilizarse.

- **Dimensionamiento de la fuente**

En esta parte se realizar el análisis sobre el consumo de corriente. En la tabla 3.8 se detalla el consumo de corriente de los componentes electrónicos.

Tabla 3.8
Consumo de componentes electrónicos

Dispositivo	Consumo de corriente	Voltaje	Potencia W
Arduino Mega	40 mA	5 V	0.2
Sensor Ping	35 mA	5 V	0.17
Xbee S1 (Tx)	45 mA	3.3 V	0.15
Xbee S1 (Rx)	45 mA	3.3 V	0.15
Servo Turnigy x2	281.2 x2 = 562.4 mA	5 V	1.4 x 2 = 2.8
Servo HD x4	162.4 x4 = 649.6 mA	5 V	0.81 x 4 = 3.24
Servo Hitec	121.7 mA	5 V	0.61
LM2596	6 mA	12 V	0.07
LM2596	6 mA	12 V	0.07
Total	1.51 mA		7.46 W

Fuente: Elaborado por el autor

Dentro del capítulo 1 en la sección 1.10.2 Consumo eléctrico, se muestra las fórmulas utilizadas para calcular la batería a utilizar. El valor de 7.46 W es utilizado para calcular el valor de la batería en amperios hora (Ah) mediante los siguientes parámetros:

a) Tiempo de autonomía = 2 horas
b) Capacidad restante en la batería = 25%
c) Voltaje de la batería =14.8 voltios

$$\boldsymbol{C = Pc * t} \quad \text{Ecuación 4}$$

$$C = 7.46\ W * 2\ h$$

$$C = 14.92\ Wh$$

$$\boldsymbol{P_{D\ (25\%)} = C * (0.25)} \quad \text{Ecuación 5}$$

$$P_{D\ (25\%)} = 14.92Wh * (0.25)$$

$$P_{D\ (25\%)} = 3.73\ Wh$$

$$\boldsymbol{P_R = C + P_{D\ (25\%)}} \quad \text{Ecuación 6}$$

$$P_R = 14.92\ Wh + 3.73\ Wh$$

$$P_R = 18.65\ Wh$$

$$\boldsymbol{C(Bat) = \frac{PR}{V}} \quad \text{Ecuación 7}$$

$$C(Bat) = \frac{18.65\ W}{14.8\ V}$$

$$C(Bat) = 1.26Ah$$

Según los cálculos realizados se requiere de una batería de 1.26 amperios hora (Ah) con un voltaje de 14.8 voltios. Según los parámetros definidos se debe utilizar una batería con un amperaje similar o superior. Por este motivo se ha seleccionado una batería de LiPo de 3 amperios hora (Ah) ya que suministra el voltaje requerido y porque cuenta con mayor rango de duración a diferencia de las baterías normales.

- **Configuración del cargador de baterías**

Cuando el nivel de energía de la batería se agota es necesario utilizar un cargador adecuado ya que las baterías de LiPo son bastante delicadas, por este motivo no pueden ser alimentadas como las otras baterías. Es por esto que se utiliza el dispositivo iMAX B6, el cual permite cargar las baterías de LiPo.

En el internet se puede encontrar diversos manuales y páginas web para la configuración del iMAX B6. En este caso se puede utilizar la página web llamada *Configurador de carga de baterías*. (Apuntes de aeromodelismo, 2017). En donde se puede conocer los parámetros necesarios para poder cargar la batería de forma adecuada. En la figura 3.13, se muestra los parámetros que se deben configurar.

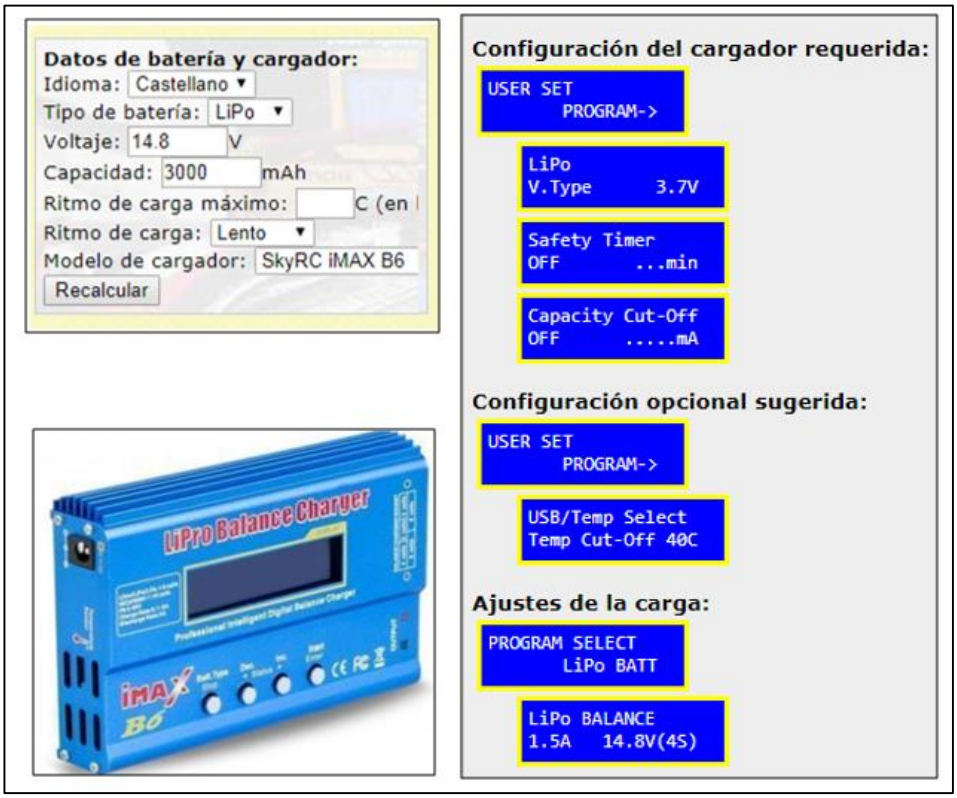


Figura 3.13: Parámetros para el cargador
Fuente: (Apuntes de aeromodelismo, 2017)

En la parte superior derecha de la imagen se observa una ventana en donde se ingresan los datos de la batería de LiPo, se ingresa estos datos y se presiona recalcular. Con esto aparece automáticamente la ventana de la izquierda con los parámetros de configuración de color azul que se deben ingresar en el cargador iMAX.

3.4 Diagrama general del software

El diagrama de flujo para el funcionamiento del robot está dividido en dos partes: el programa principal y la interfaz de usuario.

3.4.1 Programa principal

En la figura 3.14 se describe el funcionamiento del robot al recibir las ordenes enviadas desde el interfaz de usuario. Los datos al ser receptados son procesados por el Arduino Mega y este a su vez ejecutar las ordenas enviadas.

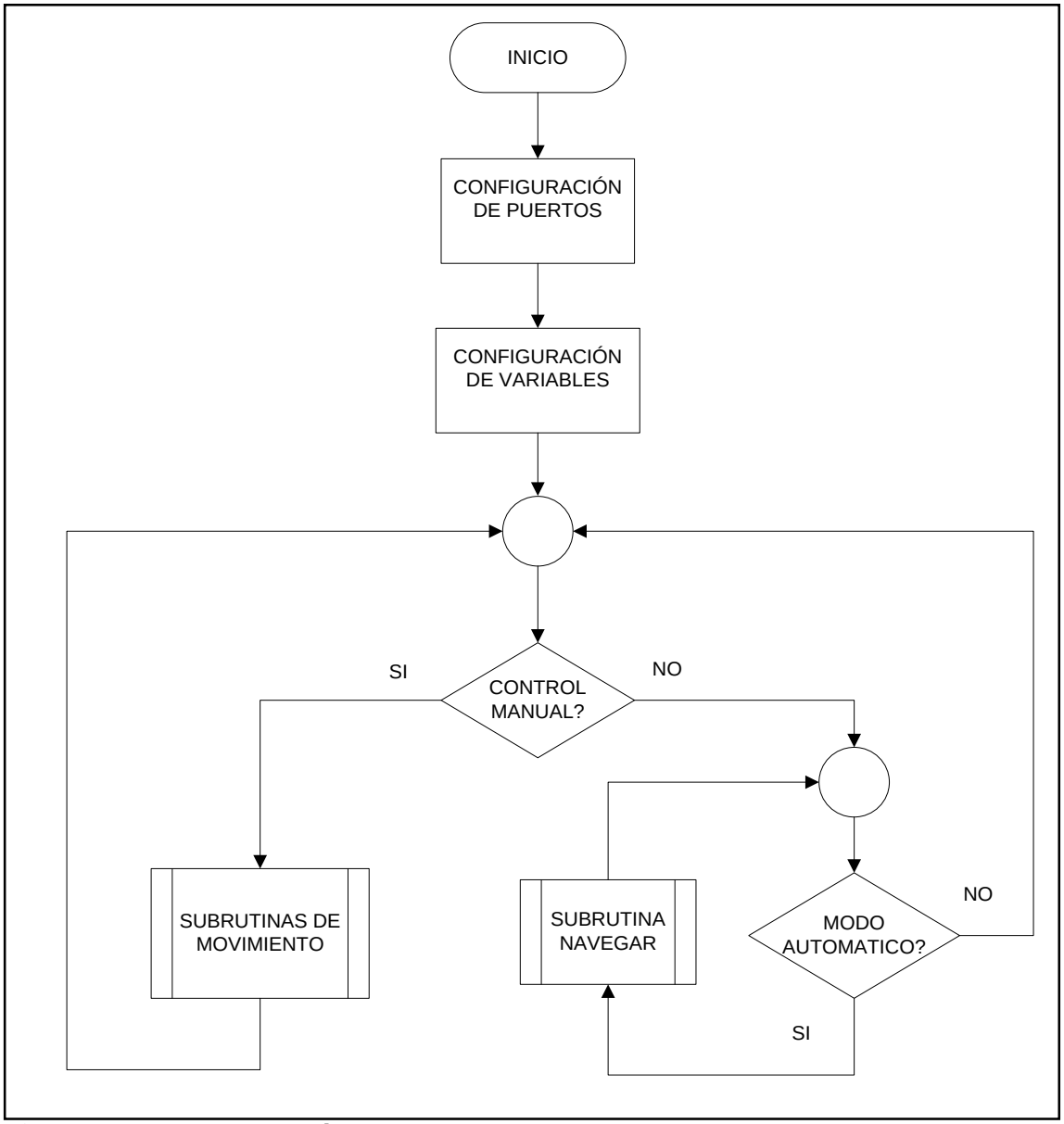


Figura 3.14: Programa principal
Fuente: El autor

3.4.2 Subrutina adelante

En la figura 3.15 se muestra el diagrama que se utiliza para ejecutar el desplazamiento hacia adelante. Esta función empieza cuando se da la orden desde la interfaz.

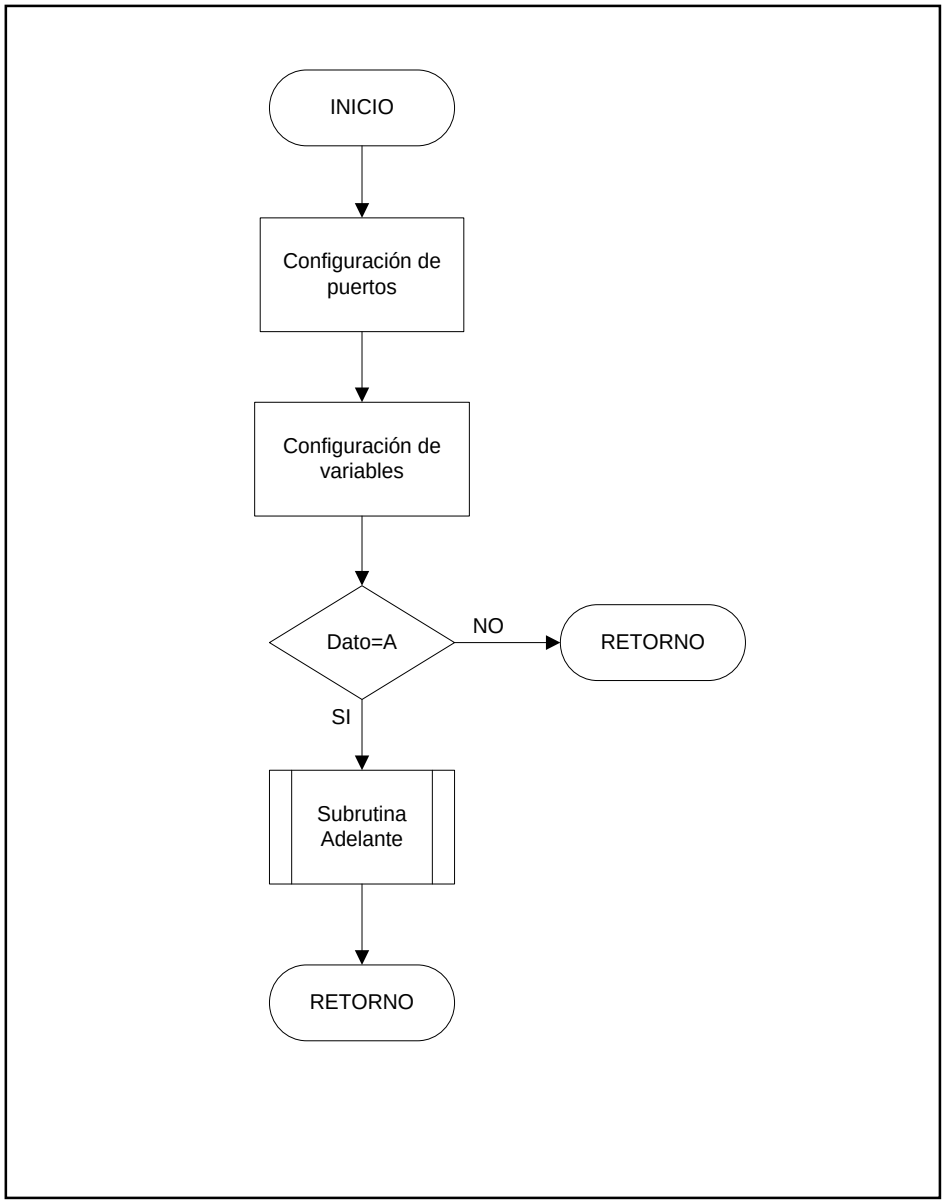


Figura 3.15: Subrutina adelante
Fuente: El autor

3.4.3 Subrutina Izquierda

En la figura 3.16 se muestra el diagrama que se utiliza para ejecutar el desplazamiento hacia la izquierda.

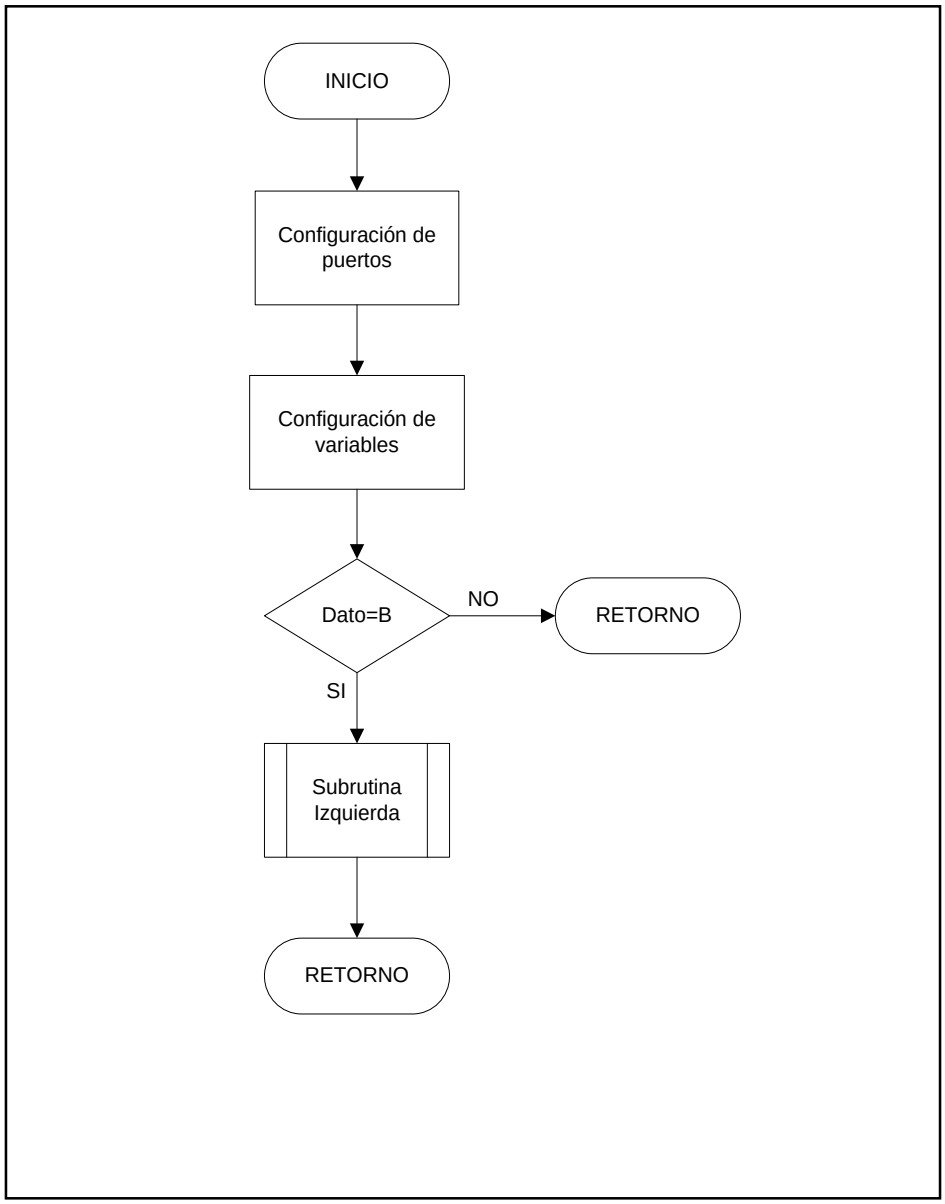


Figura 3.16: Subrutina Izquierda
Fuente: El autor

3.4.4 Subrutina Derecha

En la figura 3.17 se muestra el diagrama que se utiliza para ejecutar el desplazamiento hacia la derecha.

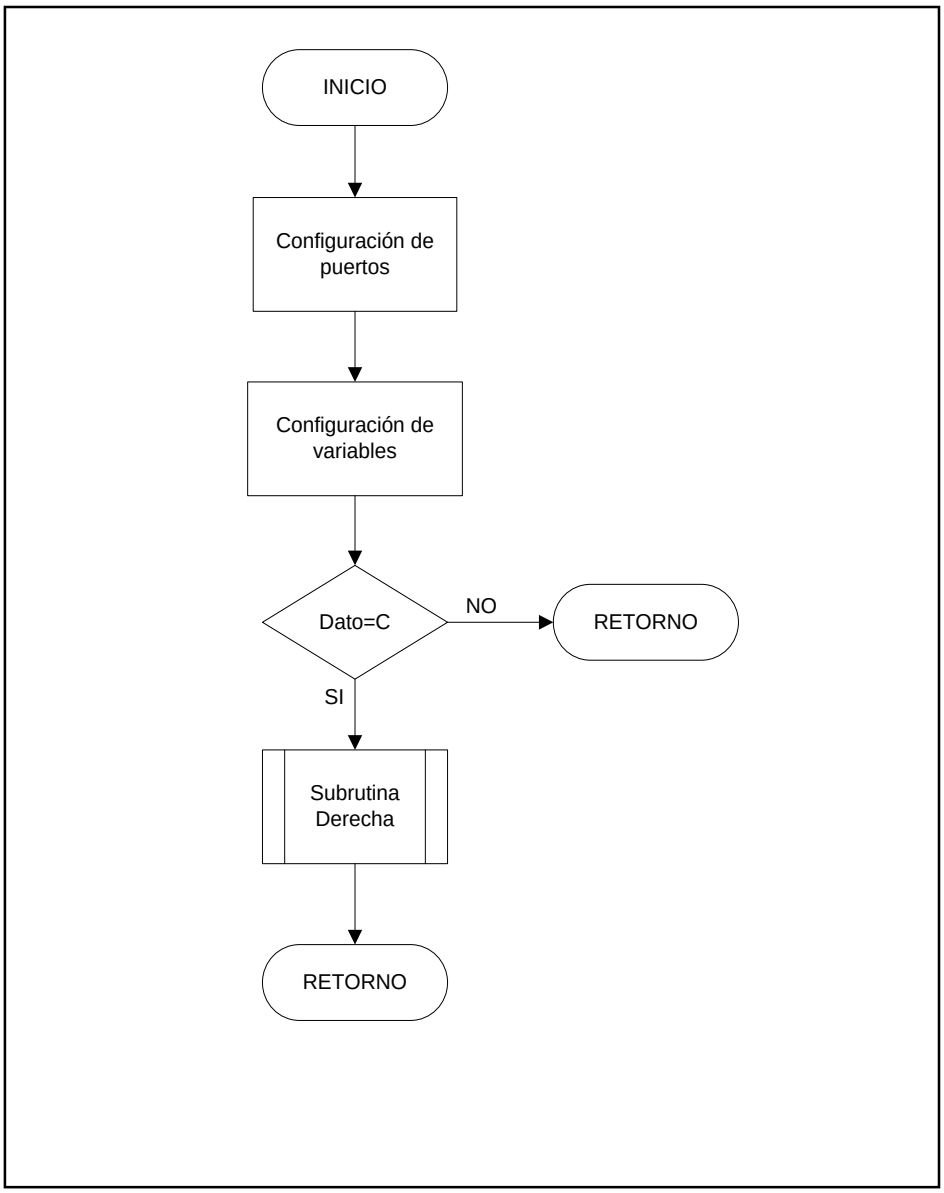


Figura 3.17: Subrutina derecha
Fuente: El autor

3.4.5 Programa para la interfaz de usuario

En la figura 3.18 se muestra el diagrama para controlar desde una computadora los movimientos del robot. Para su diseño se utiliza el software Visual Studio del cual se utiliza el lenguaje de programación Visual Basic.

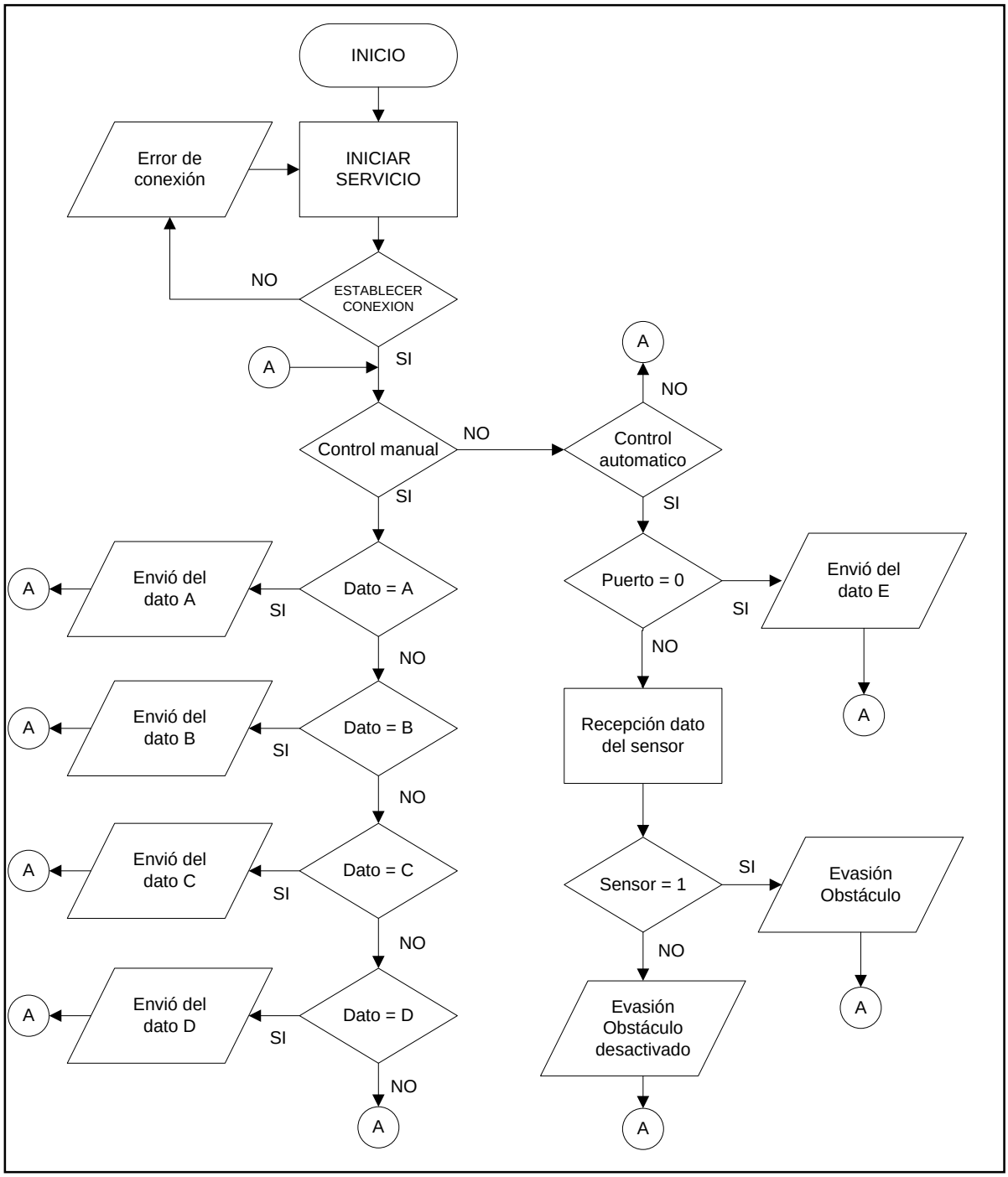


Figura 3.18: Programa para la interfaz de usuario
Fuente: El autor

3.4.6 Software empleado

En el desarrollo de este proyecto se realizan; esquemas, simulaciones y aplicaciones, por lo cual se debe utilizar el software adecuado y se los describen a continuación.

- **Arduino**

Se ha escogido este software porque es compatible con cualquier placa Arduino ya que en la realización de este proyecto se utiliza un Arduino Mega. El software Arduino es muy utilizado para el desarrollo de proyectos de electrónica y está basado en un sistema de código abierto lo que facilita el desarrollo de algoritmos. En el Anexo 5 se muestra como aprender a programar en este software.

- **Proteus**

Para el diseño electrónico de este proyecto se utilizará el software Proteus, porque permite diseñar los diagramas de la parte electrónica y la elaboración del diagrama del circuito impreso. Además, se lo utiliza porque permite la simulación de microcontroladores, enrutado manual, enrutado automático y visualización de imágenes en tres dimensiones.

- **Visual Studio**

Se utiliza Visual Studio porque es compatible con el lenguaje de programación Visual Basic.NET, el cual facilita el desarrollo de aplicaciones para la rama de la electrónica, con este software se diseñará la interfaz de usuario para el control del robot. En el Anexo 6 se muestra como aprender a programar en este software

- **XCTU**

Se utiliza el software XCTU porque es una aplicación diseñada para el manejo de módulos Xbee y porque permite la configuración y prueba de los módulos de radio frecuencia mediante el enlace en serie. En este proyecto se emplean dos módulos Xbee por lo tanto se requiere esta interface para establecer la comunicación inalámbrica.

- **AutoCAD**

Se utiliza este software porque se requiere diseñar las piezas del prototipo en dos dimensiones (2D), entre estas: Multipropósito, tipo C, tipo L, tipo pie, cintura y tipo carcaza.

- **Solidworks**

Se utiliza software para elaborar los planos de ensamblaje de las piezas que componen el robot bípedo. Estos planos son desarrollados en tres dimensiones (3D).

3.5 Aspectos técnicos del proyecto

Se ha citado previamente los componentes a utilizarse para la fabricación del robot en resumen se menciona los elementos electrónicos y mecánicos a continuación:

1) Arduino Mega (Datasheet Anexo 7)
2) Módulo Xbee (Datasheet Anexo 8)
2) Convertidor de voltaje LM2596 (Datasheet Anexo 9)
7) Servomotores (Datasheet Anexo 10)
1) Sensor de detección Ping (Datasheet Anexo 11)
1) Lamina Tol galvanizado
1) Batería de LiPo

Una vez definido los componentes se procede a de enlistar las características técnicas.

- Dimensiones aproximadas: 15cm x 23cm x 15 cm
- Peso aproximado a 1.55 kg
- Peso aproximado sin batería (la batería si puede ser desmontada) 1.25 kg
- Movimientos a realizar: adelante, izquierda y derecha
- Desplazamiento mediante la caminata bípeda
- Articulado mediante 7 servomotores

3.6 Análisis de costos

En esta parte se realiza el análisis de costos para la realización del proyecto. En la tabla 3.9 se muestra los costos referenciales de los elementos utilizados para la fabricación del robot bípedo.

Tabla 3.9
Costos de los materiales

Cant.	Descripción materiales	Costo I.	Costo total	% aporte	Costo
2	Servo Turnigy	29,22	58,44	100%	58,44
4	Servo Power HD	20,98	83,92	100%	83,92
1	Servo Hitec	15,00	15,00	100%	15,00
2	Regulador Lm2596	3,50	7,00	100%	7,00
2	Xbee	35,00	70,00	100%	70,00
1	Arduino Mega2560	24,50	24,50	100%	24,50
1	Sensor ping parallax	50,90	50,90	100%	50,90
1	Tol galvanizado	10,00	10,00	100%	10,00
1	Xbee explorer	8,50	8,50	100%	8,50
1	Capacitor 104 (0,1uF)	0,10	0,10	100%	0,10
1	Capacitor 100uF	0,18	0,18	100%	0,18
2	Capacitor 22pF	0,10	0,20	100%	0,20
1	Diodo led	0,10	0,10	100%	0,10
2	Resistencia	0.04	0.08	100%	0.08
2	Espadín macho	0,50	1,00	100%	1,00
4	Borneras 2 pines	0,28	1,12	100%	1,12
1	Interruptor de placa	0.35	0.35	100%	0.35
1	Interruptor normal	0,65	0,65	100%	0,65
1	Batería de LiPo	50,00	50,00	100%	50,00
1	Pintura, Pernos y remaches	60,00	60,00	100%	60,00
	Subtotal				442,04
	+ IVA (12%)		53.04		495,08
	Imprevisto (5%)		24.75		**519,83**

Fuente: Elaborado por el autor

Dentro del análisis de costos también es necesario mencionar el costo operativo de la mano de obra, el cual se lo ha estimado utilizando el salario básico unificado que para el año 2019 según el incremento realizado es de $394 dólares mensuales por una jornada de 160 horas laborables.

USD 394/160 horas = $ 2,46 por hora

Con este valor se realiza en la tabla 3.10 el costo de la mano de obra.

Tabla 3.10
Costo mano de obra

HORAS/DIAS	DIAS A LA SEMANA	SEMANAS	COSTO HORA	TOTAL
2	5	18	$ 2,46	442,8

Fuente: elaborado por el autor

Además, se incluye la parte de gastos imprevistos, como condición de diseño, se aplica un 10% adicional al costo de mano de obra por situaciones diversas. Por este motivo se añade la cantidad de USD 44.28. En la tabla 3.11 se muestra el costo total para la elaboración del robot bípedo.

Tabla 3.11
Egresos

DESCRIPCIÓN	TOTAL
Materiales	519,83
Costo mano de obra	442,8
Imprevistos	44,28
Total presupuesto del proyecto	1006,91

Fuente: elaborado por el autor

3.7 Ventajas

- Los materiales usados para el desarrollo de este trabajo son de fácil acceso y de bajos costos en el mercado nacional.

- Debido a la estructura del tol, el robot bípedo es de peso ligero.

- Control mediante conectividad inalámbrica.

- Estructurado con 7 grados de libertad para una mejor movilidad.

- Basado en tecnología Arduino porque es fácil de usar aun teniendo pocos conocimientos de programación, es software libre y tiene un bajo costo.

- La operatividad del robot bípedo mediante el uso de la interfaz gráfica es amigable con el usuario.

- Presenta el parámetro de estabilidad ya que se utilizará servomotores con un alto rango de fuerza, esto le permite mantener el equilibrio en los movimientos a realizar.

CAPITULO 4

IMPLEMENTACIÓN

En este capítulo se describe el proceso que se debe realizar para el acople de las piezas metálicas del robot bípedo y también se describe las pruebas realizadas para verificar el correcto funcionamiento.

4.1 Diseño del circuito de control

En esta parte se describe los pasos para la elaboración del circuito electrónico, para esto se debe determinar los pines de conexión del Arduino Mega a utilizar. Estos parámetros son utilizados para el diseño del esquema general del circuito.

4.1.1 Conexiones del Arduino Mega

Las conexiones del Arduino Mega se utilizan para controlar los componentes electrónicos de la tarjeta. A continuación, se presentan los pines utilizados para este proyecto.

- Los pines 5,6,7,8,9,10 y 11 han sido configurados como salidas de datos, su función es controlar el movimiento de los servomotores.

- El pin 14 está configurado como entrada y salida, su función es enviar y recibir pulsos lógicos para que el sensor trabaje correctamente.

- El pin 13 se utiliza para visualizar el encendido y apagado de un diodo led. Este componente permite verificar el funcionamiento del sensor cuando detecta un objeto.

- Los pines 16 y 17 se utilizan para establecer la comunicación serial entre el Arduino Mega y el módulo Xbee.

- El pin 3V3 suministra un voltaje de 3,3 voltios, el cual se utiliza para alimentar al módulo Xbee.

- El pin VIN es una entrada que permite alimentar al Arduino Mega, este pin se utiliza para conectar la salida del módulo reductor de voltaje Step Down.

En la tabla 4.1 se observa los recursos utilizados del Arduino Mega según sus pines y su función.

Tabla 4.1
Pines utilizados del Arduino Mega

Pin	Componente	Función	Designación
5	Servo 1	Tobillo izquierdo	Salida
6	Servo 2	Tobillo derecho	Salida
7	Servo 3	Rodilla izquierda	Salida
8	Servo 4	Rodilla derecha	Salida
9	Servo 5	Pierna izquierda	Salida
10	Servo 6	Pierna derecha	Salida
11	Servo 7	Cabeza	Salida
14	Sensor Ping	Detección	E/S
13	Diodo led	Indicador	Salida
16	TX2	Transmisor	Entrada
17	Rx2	Receptor	Entrada
3V3	Voltaje 3 voltios	Fuente para Xbee	Entrada
GND	Tierra	Referencia 0V	Salida
VIN	Voltaje entrada	Alimentación 5V	Entrada

Fuente: Elaborado por el autor

4.1.2 Diagrama esquemático del circuito electrónico

En la siguiente figura 4.1 se muestra el diagrama esquemático del circuito electrónico para el control del robot bípedo.

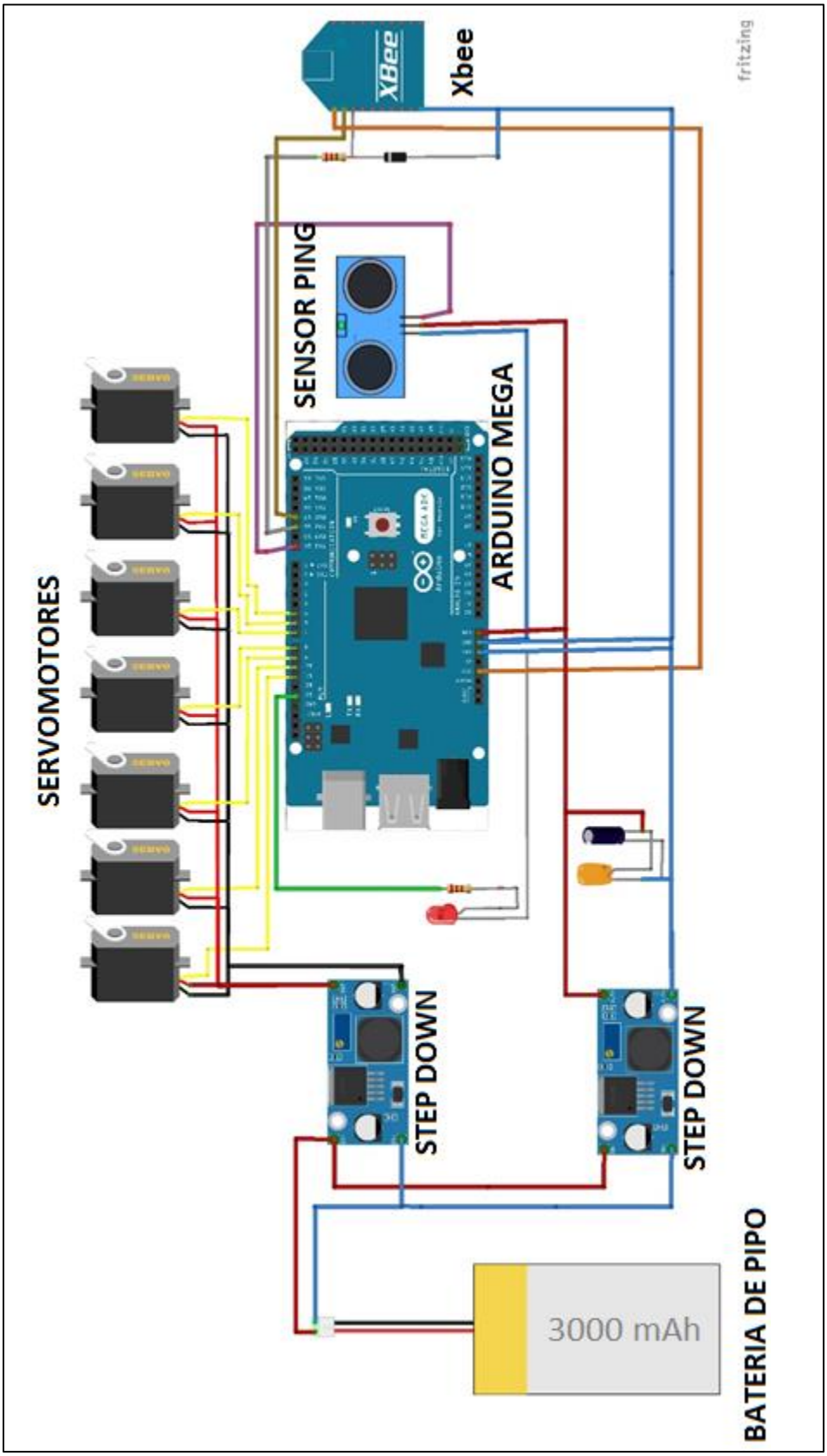


Figura 4.1: Diagrama esquemático general
Fuente: Elaborado por el autor

4.1.3 Elaboración del circuito electrónico

El diseño del circuito electrónico es realizado mediante el software Proteus. En esta plataforma se diseña la placa de circuito impreso (PCB). En la figura 4.2 se muestra el esquema de pistas electrónicas.

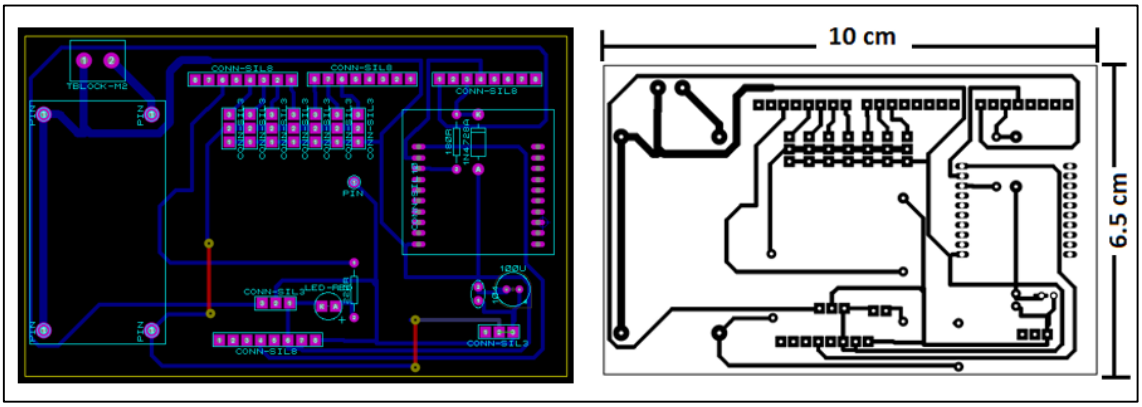


Figura 4.2: Diagrama de pistas electrónicas
Fuente: Elaborado por el autor

El diagrama obtenido prosigue a ser impreso en una hoja de papel termo transferible con las siguientes medidas: 10cm de ancho y 6.5 cm de alto. Después se realizar el proceso de transferencia hacia la baquelita para esto se ha seguido los siguientes pasos.

a) Cortar la baquelita de cobre según las dimensiones del circuito impreso.
b) Sujetar la baquelita de cobre contra el papel del circuito impreso.
c) Introducir la baquelita de cobre en un recipiente que contenga ácido férrico y agua.
d) Agitar continuamente hasta obtener las pistas como se muestra en la figura 4.3.

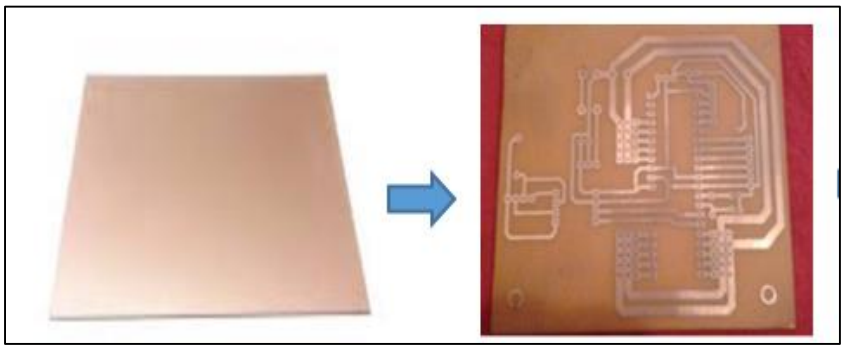


Figura 4.3: Circuito impreso en baquelita
Fuente: Elaborado por el autor

Posteriormente se realiza las perforaciones con el taladro y después se suelda los componentes electrónicos a la placa mediante el cautín y el estaño. En la figura 4.4 se muestra las pistas soldadas.

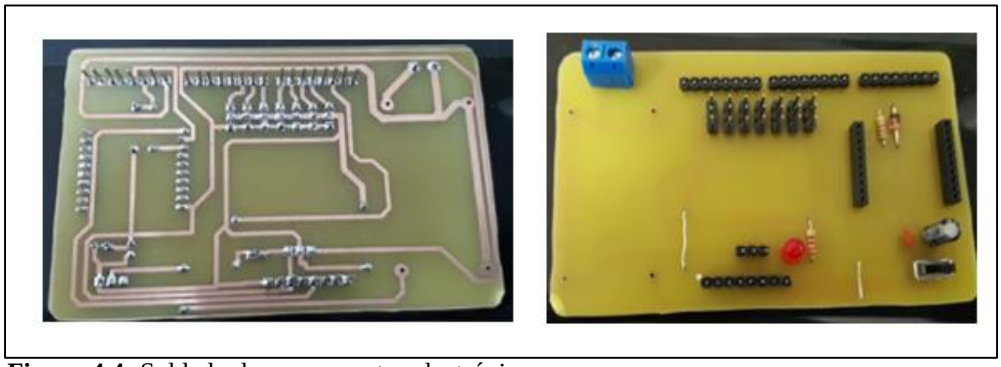


Figura 4.4: Soldado de componentes electrónicos
Fuente: Elaborado por el autor

En la figura 4.5 se muestra la placa terminada con las dimensiones 10cm x 6.5cm. En ella se puede observar los conectores para los componentes electrónicos a utilizar.

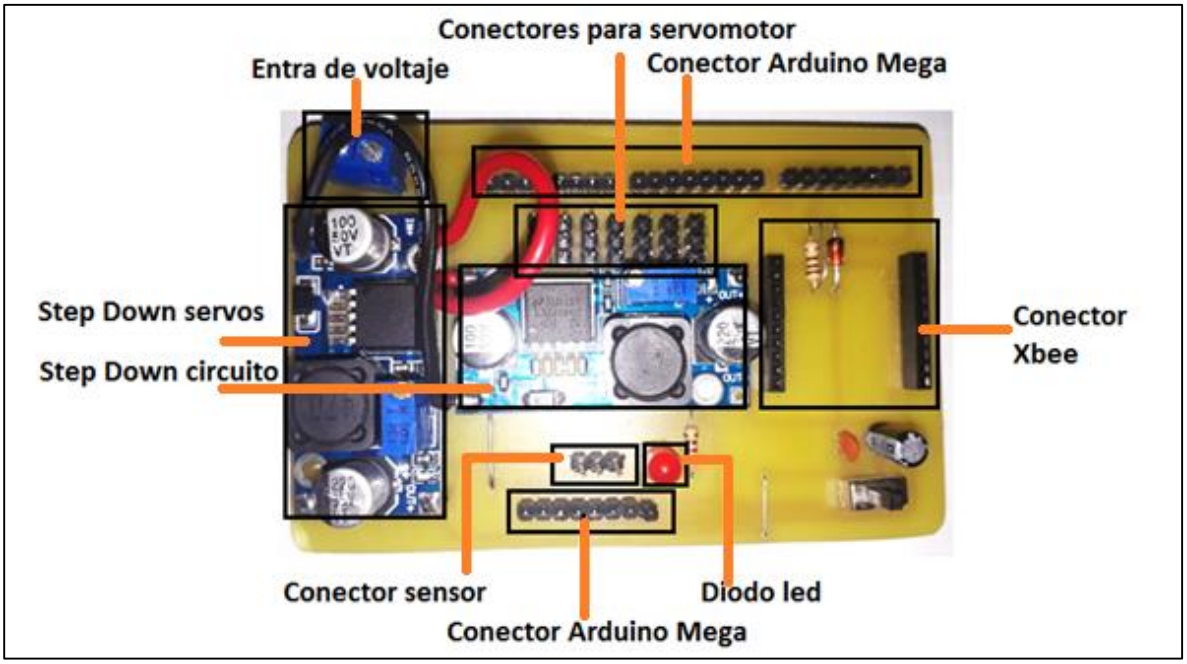


Figura 4.5: Placa electrónica terminada
Fuente: Elaborado por el autor

4.2 Ensamblaje del robot

El ensamblaje del robot bípedo es realizado mediante el acople de piezas metálicas descritas en la propuesta. Estas piezas son colocadas en un orden determinado para lo cual se debe seguir el orden establecido en los planos.

4.2.1 Planos para el ensamblaje

El ensamblaje de las piezas del robot ha sido diseñado mediante el software Solidworks, a continuación, se describe el proceso para el acople.

- **Tobillo**

En las figuras 4.6, 4.7 y 4.8 se muestra paso a paso el ensamblaje de las piezas para el armado del tobillo.

- **Rodilla**

En las figuras 4.9 y 4.10 se muestra el ensamblaje de las piezas para el armado de la rodilla.

- **Pierna**

En las figuras 4.11 y 4.12 se muestra el ensamblaje de las piezas para formar la pierna del robot.

- **Cadera**

En la figura 4.13 se muestra el ensamblaje de piezas para formar la cadera del robot.

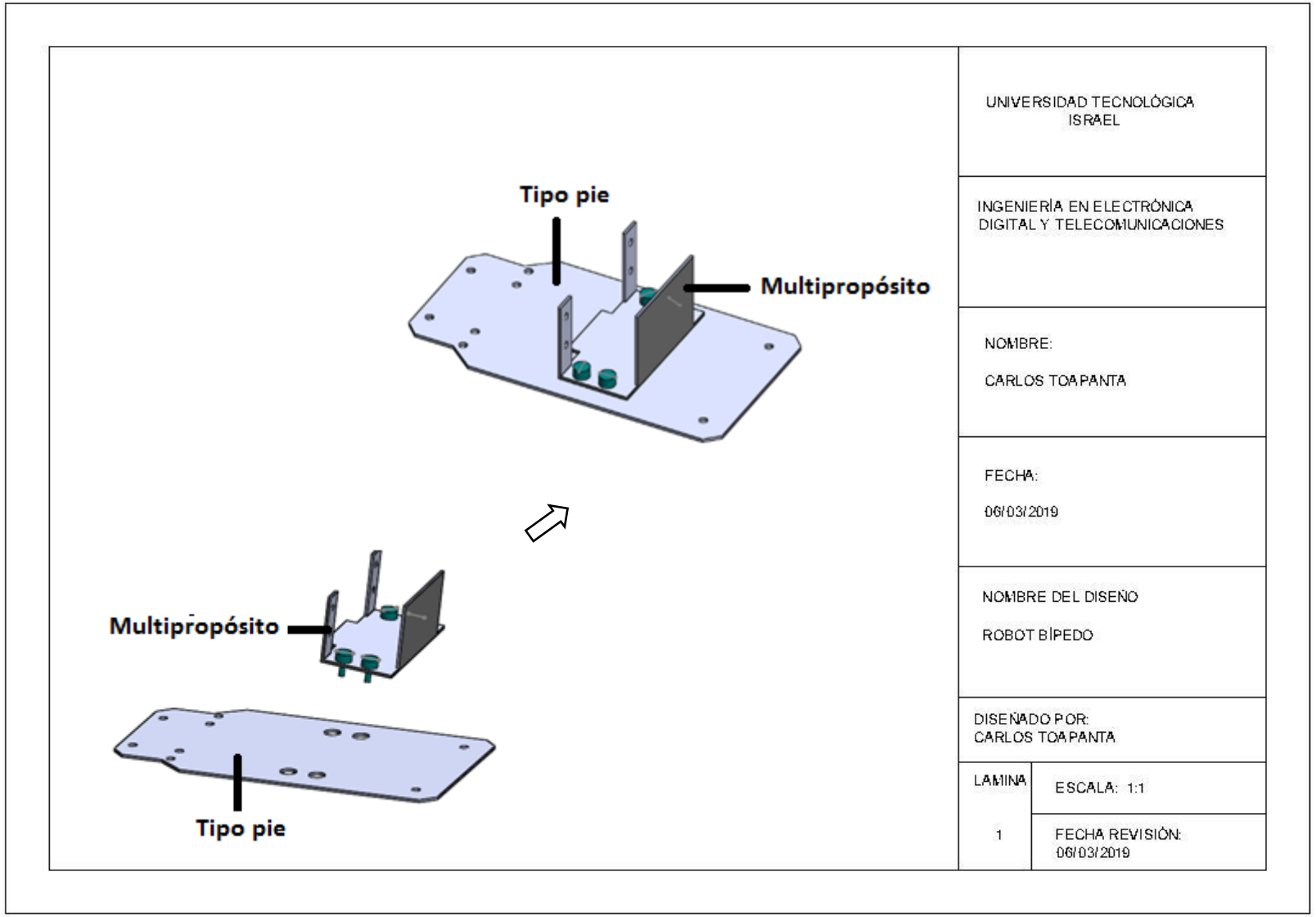


Figura 4.6: Armado del tobillo primera parte
Fuente: Elaborado por el autor

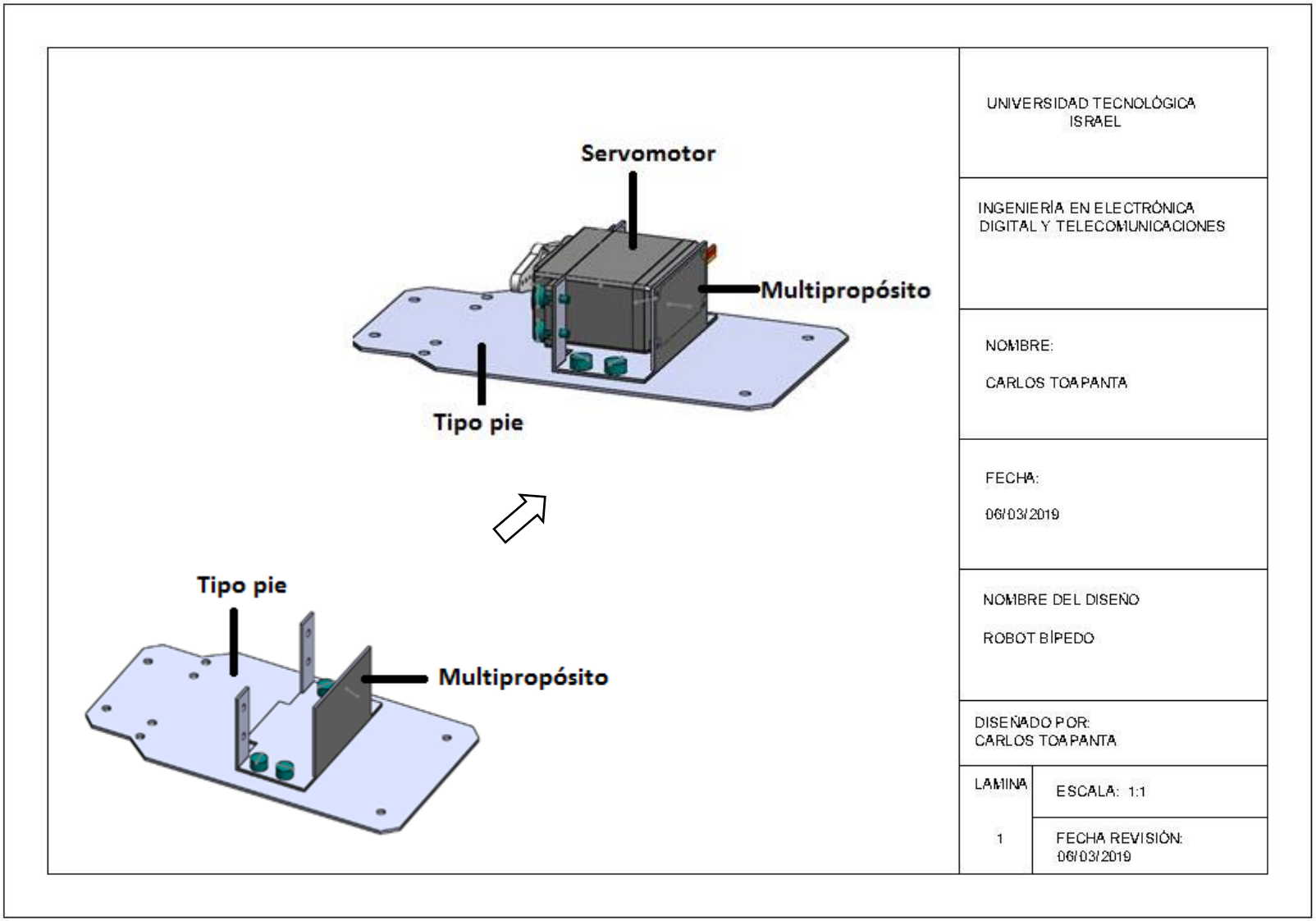


Figura 4.7: Armado del tobillo segunda parte
Fuente: Elaborado por el autor

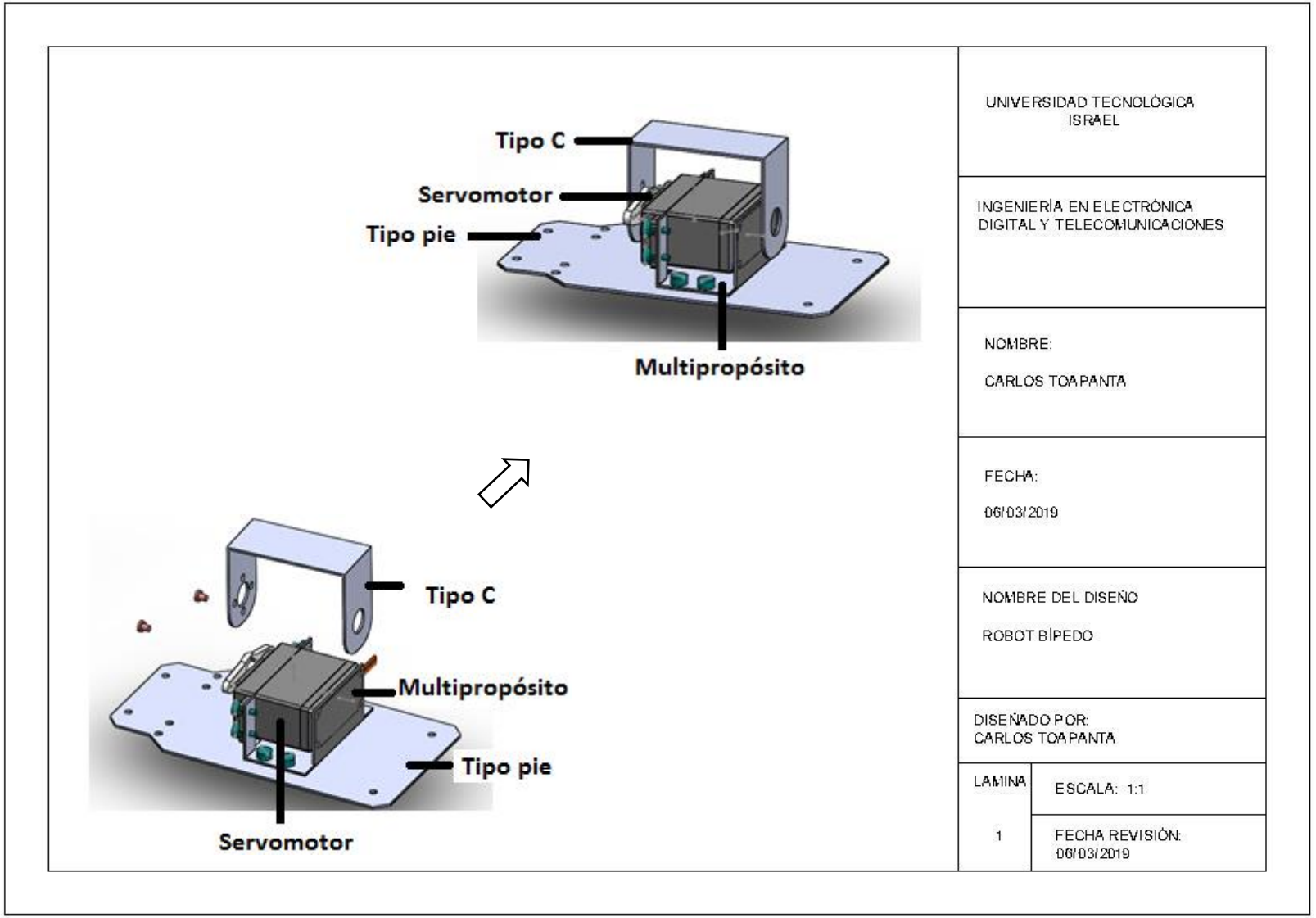


Figura 4.8: Armado del tobillo tercera parte
Fuente: Elaborado por el autor

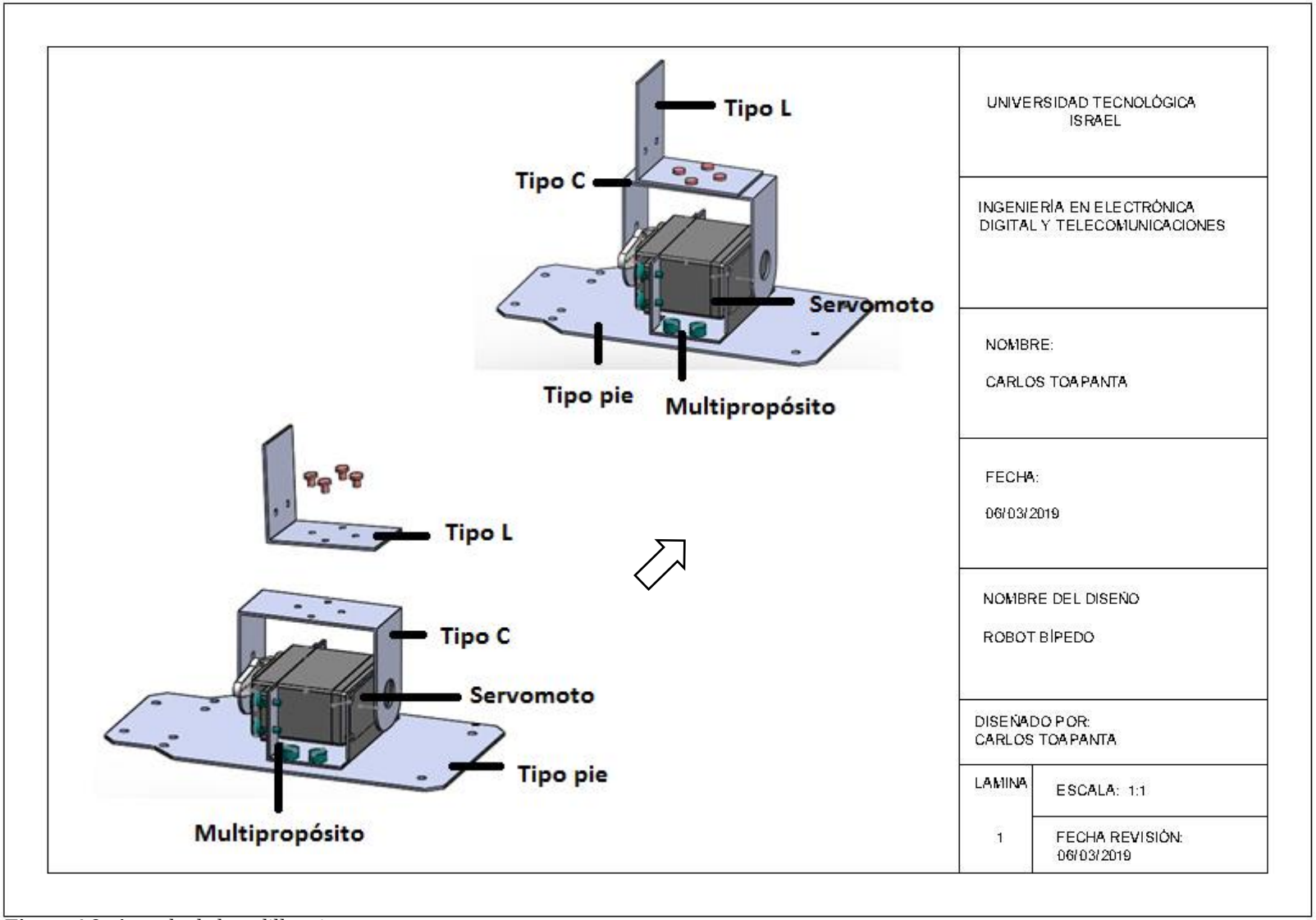


Figura 4.9: Armado de la rodilla primera parte
Fuente: Elaborado por el autor

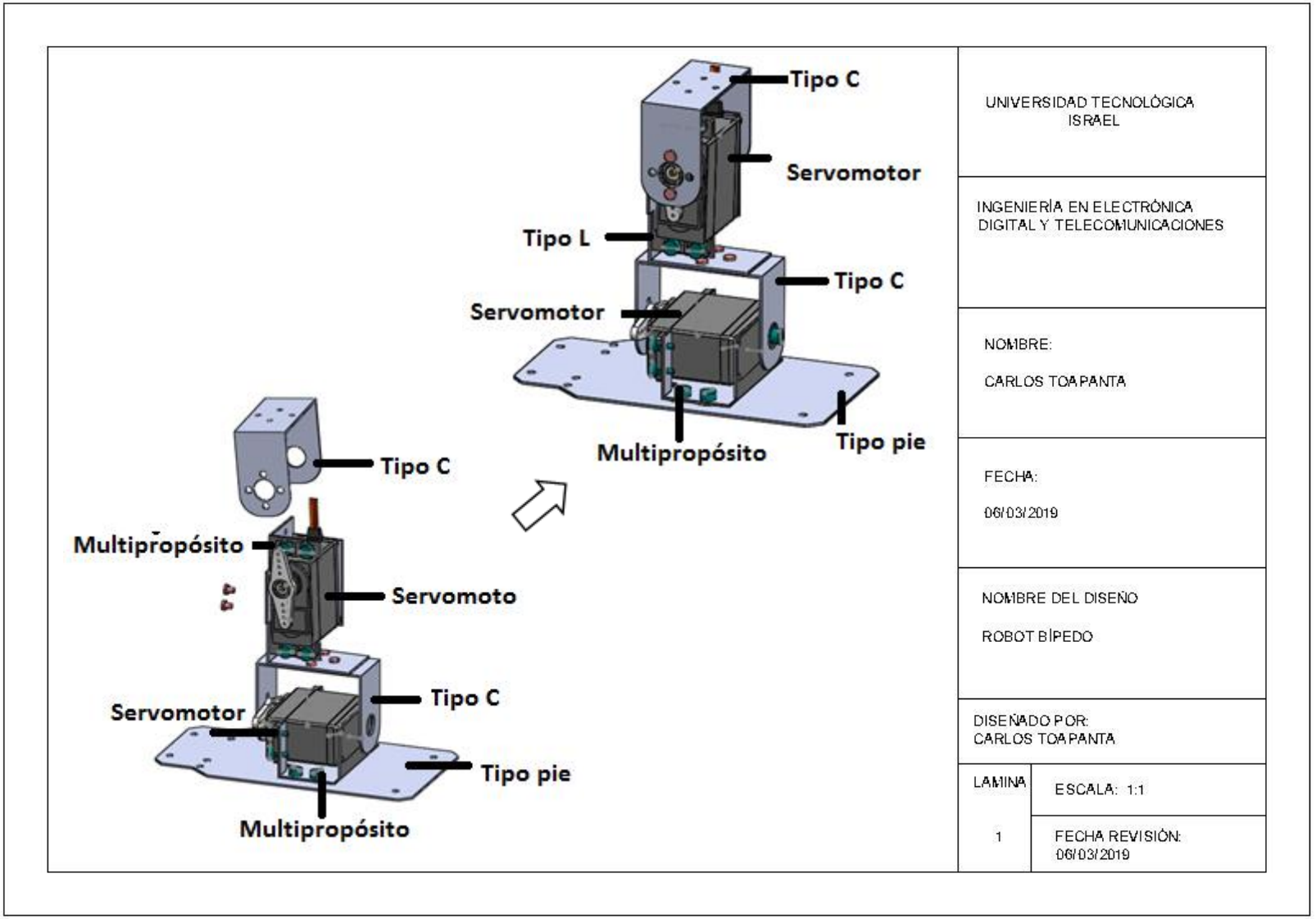


Figura 4.10: Armado de la rodilla segunda parte
Fuente: Elaborado por el autor

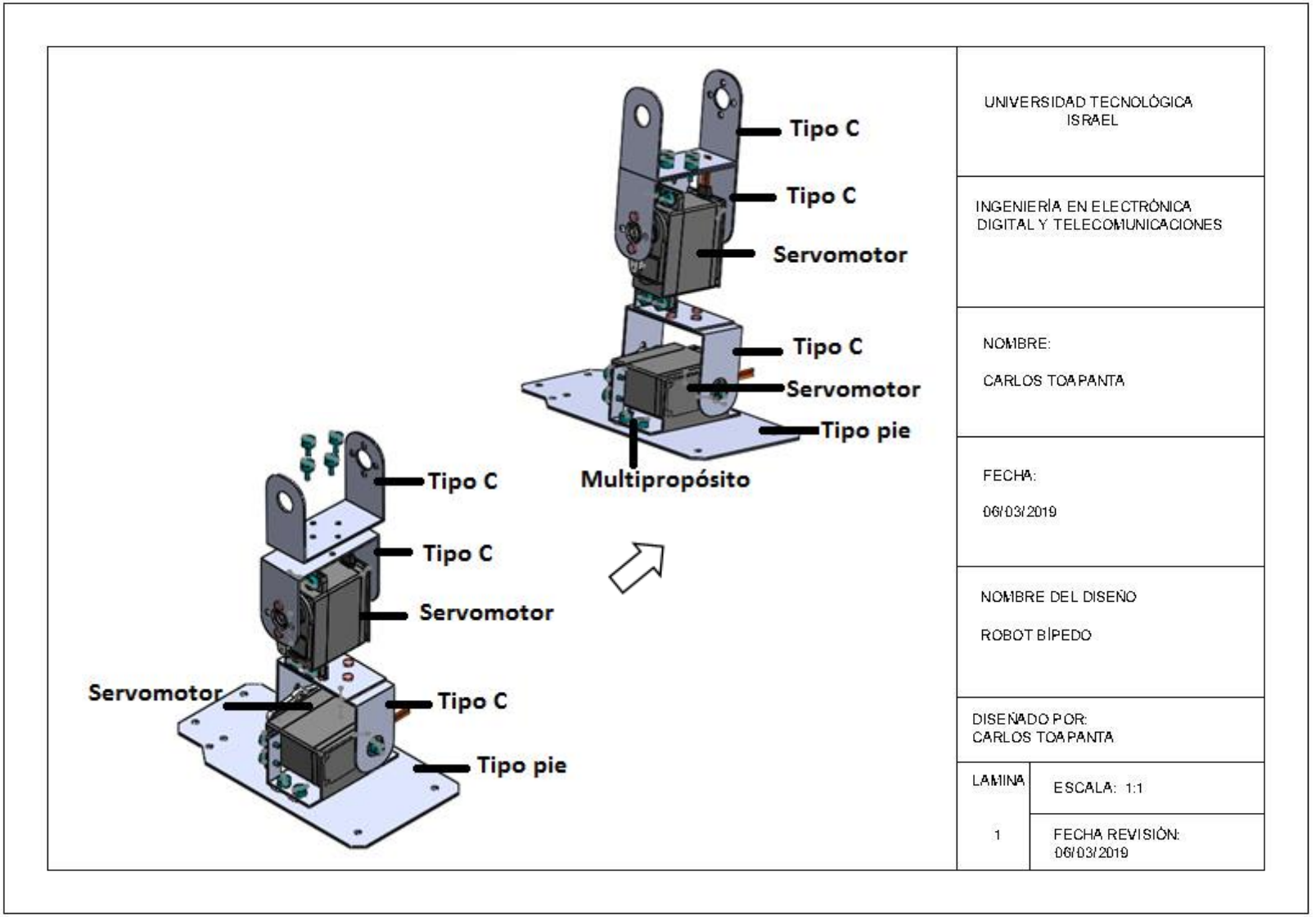


Figura 4.11: Armado de la pierna primera parte
Fuente: Elaborado por el autor

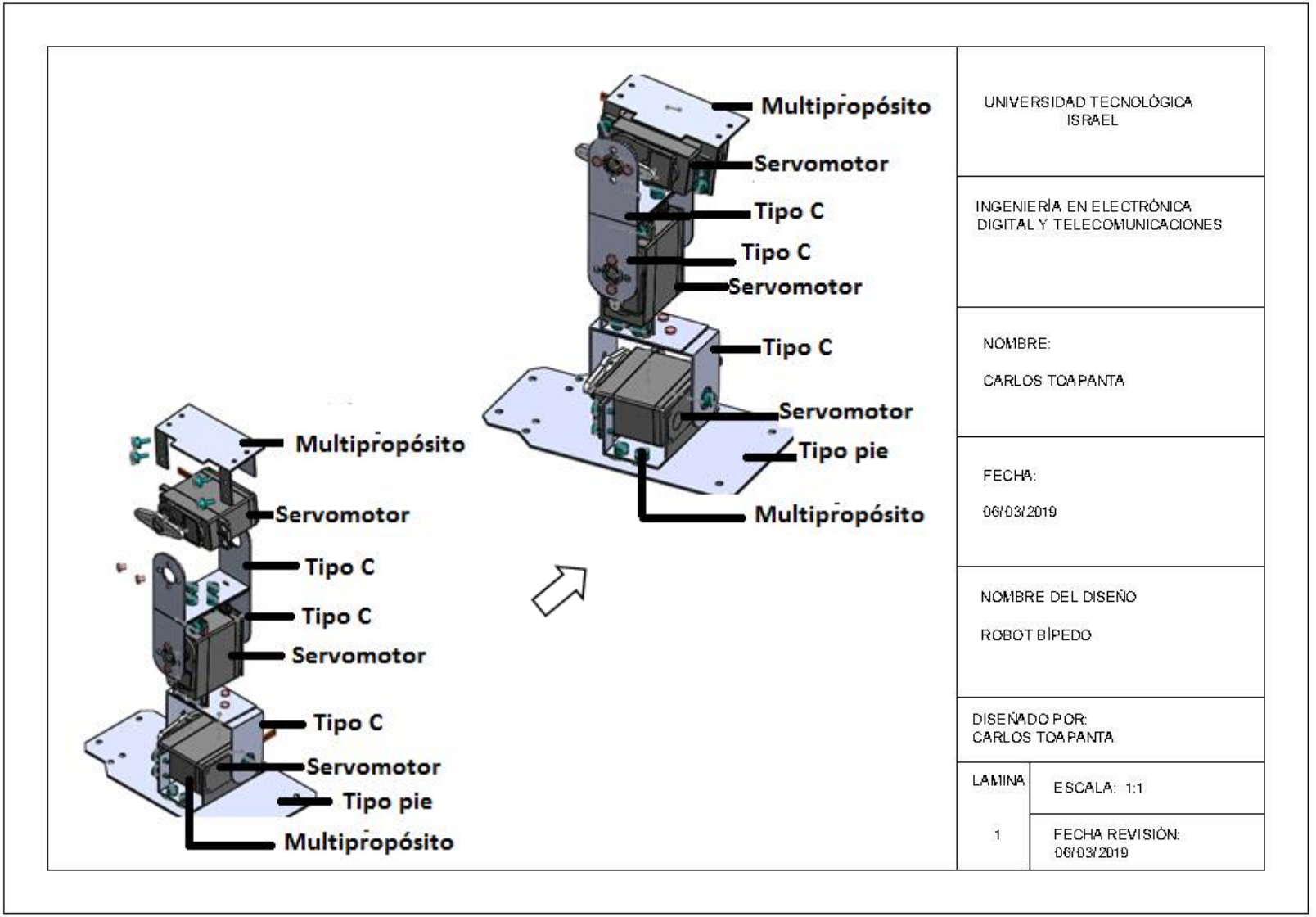


Figura 4.12: Armado de la pierna segunda parte
Fuente: Elaborado por el autor

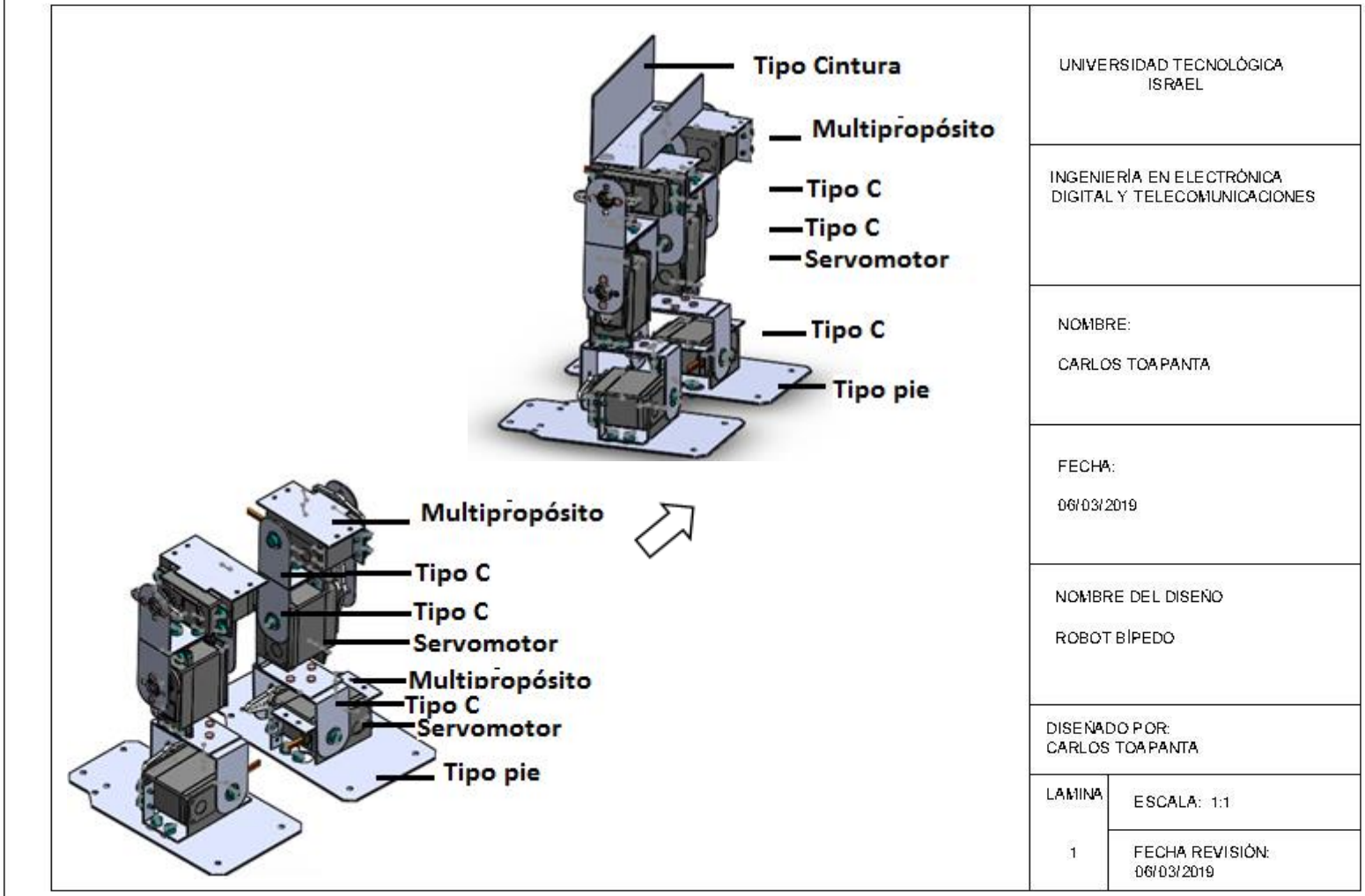


Figura 4.13: Armado de la cadera
Fuente: Elaborado por el autor

4.2.2 Acople mecánico

El acople mecánico consiste en armar las piezas metálicas y con estas formar las articulaciones para el tobillo, rodilla y pierna.

- **Ensamblaje del tobillo**

El ensamblaje del tobillo consiste en acoplar tres piezas y estas son: tipo pie, multipropósito y tipo "C". Además, se utiliza un servomotor de 15.5 kilogramos de torque para articular el movimiento. En la figura 4.14 se muestra la forma del tobillo.

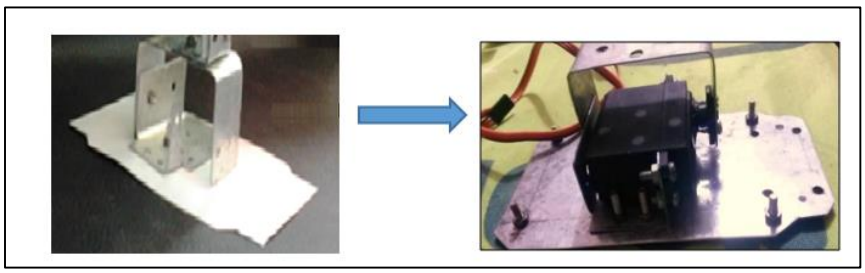


Figura 4.14: Tobillo del robot
Fuente: Elaborado por el autor

- **Ensamblaje de la rodilla**

El ensamblaje de la rodilla consiste en acoplar el tobillo ya elaborado junto a tres piezas y estas son: multipropósito, tipo "C" y tipo "L". Además, en la rodilla se ha colocado un servomotor de 8.6 kilogramo de torque. En la figura 4.15 se muestra la forma de la rodilla.

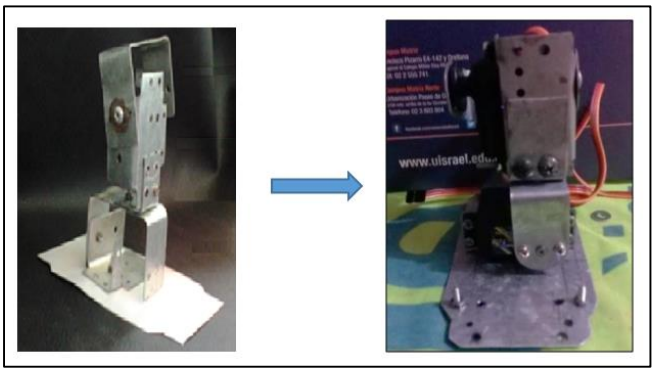


Figura 4.15: Rodilla del robot
Fuente: Elaborado por el autor

- **Ensamblaje de la pierna**

El ensamblaje de la pierna consiste en unir el tobillo y la rodilla junto a dos piezas las cuales son: multipropósito y tipo “C”. Además, se utiliza un servomotor de 8.6 kilogramos de torque para formar la articulación. En la figura 4.16 se muestra la forma de la pierna.

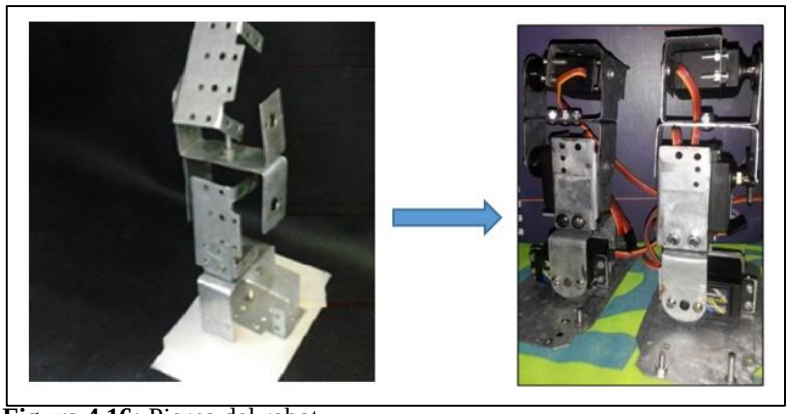


Figura 4.16: Pierna del robot
Fuente: Elaborado por el autor

- **Ensamblaje de la cintura**

El ensamblaje de la cintura consiste en acoplar las piernas mediante la pieza tipo cadera. En la figura 4.17 se muestra el robot ensamblado.

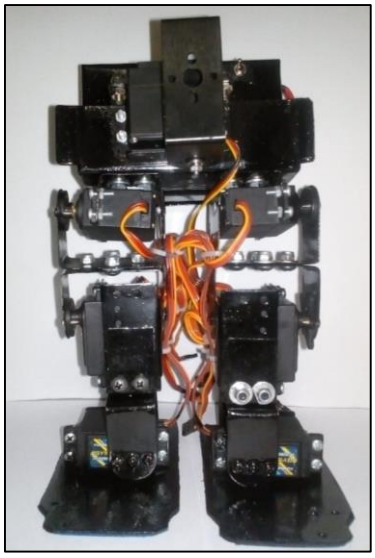


Figura 4.17: Estructura terminada
Fuente: Elaborado por el autor

4.2.3 Montaje del Sensor

El montaje consiste en colocar el sensor de detección en la parte superior del robot. Este dispositivo es el encargado de detectar los obstáculos dentro del rango predeterminado. En la figura 4.18 se muestra el sensor acoplado al robot.

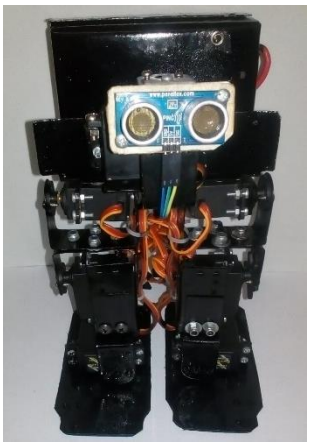


Figura 4.18: Montaje del sensor
Fuente: Elaborado por el autor

4.2.4 Acople del Arduino con la placa electrónica

Este acople consiste en colocar el circuito electrónico sobre el Arduino Mega. Se realiza esta unión porque el Arduino Mega no cuenta con los pines adecuados para la conexión directa de los componentes electrónicos a utilizar en este proyecto. Por lo cual la placa electrónica se utiliza como interfaz física. En la figura 4.19 se muestra el acople electrónico.

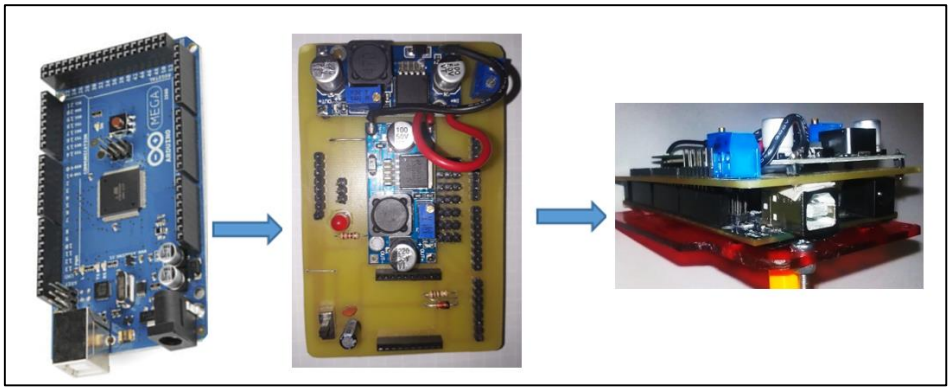


Figura 4.19: Montaje del Arduino Mega
Fuente: Elaborado por el autor

4.2.5 Montaje de la placa electrónica al robot

En esta parte se coloca el acople electrónico dentro del revestimiento metálico. En la figura 4.20 se observa la colocación del acople electrónico.

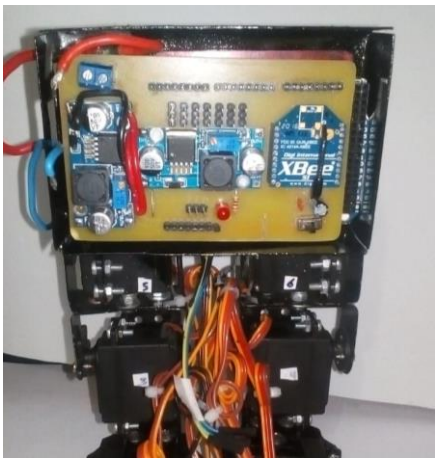


Figura 4.20: Placa acoplada al robot
Fuente: Elaborado por el autor

4.2.6 Montaje de la batería

La batería es colocada en la parte superior robot entre el sensor y el revestimiento del circuito. Es colocada en esta posición para que no afecte al equilibrio del robot. En la figura 4.21 se observa la colocación de la batería.

Figura 4.21: Montaje de la batería de LiPo
Fuente: Elaborado por el autor

4.2.7 Conexión de puertos

En esta parte se describe la conexión de los puertos de la tarjeta electrónica para la batería, los servomotores y del sensor de detección.

- **Puerto para la batería**

  El circuito cuenta con una bornera de dos entradas como se observa en la figura 4.22. Este componente permite conectar los cables de la batería, para esto se debe tomar en cuenta que el cable azul debe ir al lado izquierdo mientras que el cable rojo al lado contrario.

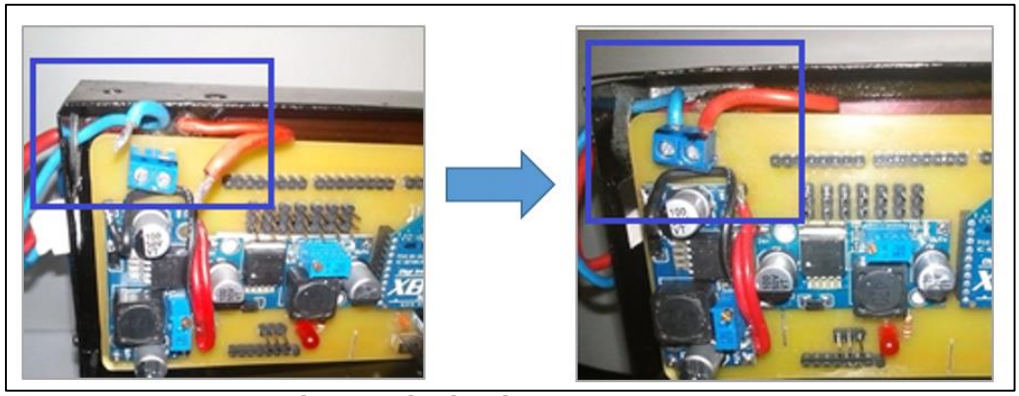


Figura 4.22: Puertos para el convertidor de voltaje
Fuente: Elaborado por el autor

- **Puerto para el sensor**

El circuito cuenta con un conector tipo espadín de tres pines, el cual se utiliza para conectar los cables del sensor como se muestra en la figura 4.23.

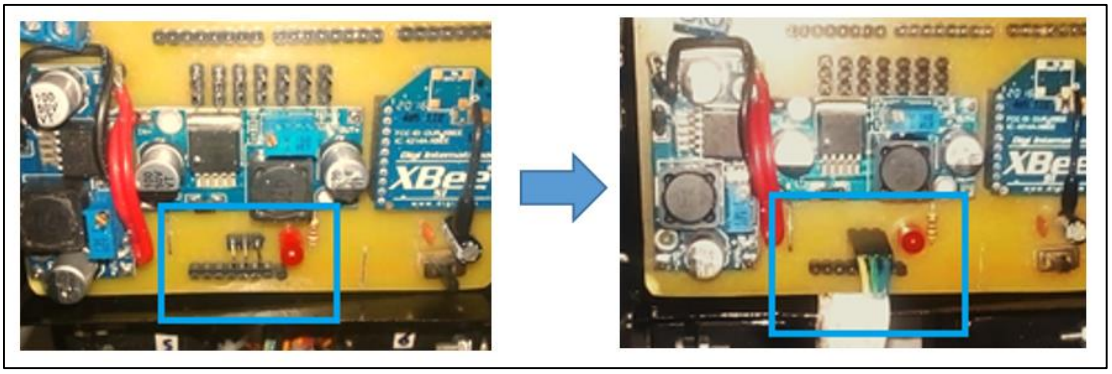


Figura 4.23: Puertos para el sensor ultrasónico
Fuente: Elaborado por el autor

- **Puertos para los servomotores**

Esta conexión se realiza mediante los pines tipo espadín que se encuentran ubicados en la parte superior del circuito. Los servomotores están enumerados del uno al siete y son colocados de derecha a izquierda para reconocer la ubicación fácilmente. En la figura 4.24 se muestra el puerto de conexión para los servomotores.

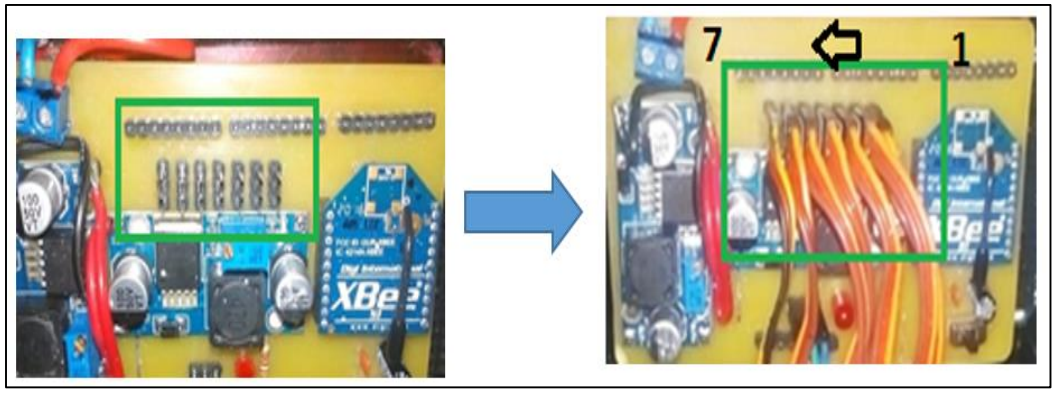


Figura 4.24: Montaje de la batería de Lipo
Fuente: Elaborado por el autor

4.3 Programación y configuración

En esta parte se menciona el funcionamiento del algoritmo de control, el funcionamiento de la interfaz de usuario y la configuración de los módulos Xbee.

4.3.1 Programación del software Arduino

El algoritmo del programa está desarrollado en lenguaje C++ de Arduino. El código está desarrollado en base al diagrama de flujo que se muestra en el capítulo 3, figura 3.14. Este algoritmo permite acciona los movimientos del robot bípedo. En el Anexo 3 se muestra el algoritmo realizado.

4.3.2 Funcionamiento de la interfaz de usuario

La interfaz de usuario realizada en Visual Studio permite controlar el desplazamiento del robot de forma inalámbrica mediante el control manual y el control automático. Para iniciar con la comunicación se debe abrir la ventana de configuración como se muestra en la figura 4.25, en donde se selecciona el puerto de comunicación y se presiona conectar.

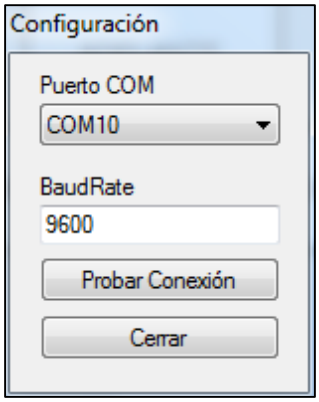


Figura 4.25: Ventana de configuración
Fuente: Elaborado por el autor

El primer modo es control manual y se muestra en la figura 4.26. Se utiliza para dirigir la caminata del robot mediante los botones de color blanco: adelante, izquierda, derecha, detener.

Figura 4.26: Ventana de control manual
Fuente: Elaborado por el autor

El segundo modo es el control automático y se muestra en la figura 4.27. Al activar este modo de operación el robot se desplaza de forma independientemente. Para esto entra en funcionamiento el sensor de detección y cuando este detecta un obstáculo el robot cambia de dirección para no chocar. El algoritmo de programación se muestra en el Anexo 4.

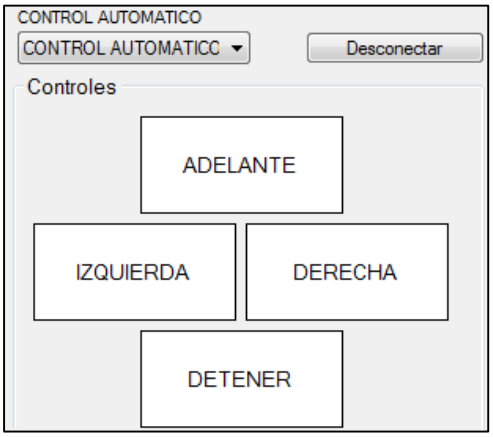


Figura 4.27: Ventana de control automático
Fuente: Elaborado por el autor

4.3.3 Configuración nodo coordinador

La configuración del módulo Xbee es realizado mediante el software XCTU. Para utilizarlo se debe conectar el módulo Xbee al módulo Explorer USB y este a su vez al puerto serial de la computadora. En la figura 4.28 se observa la pantalla inicial del software en la cual se debe seleccionar la opción Add devices.

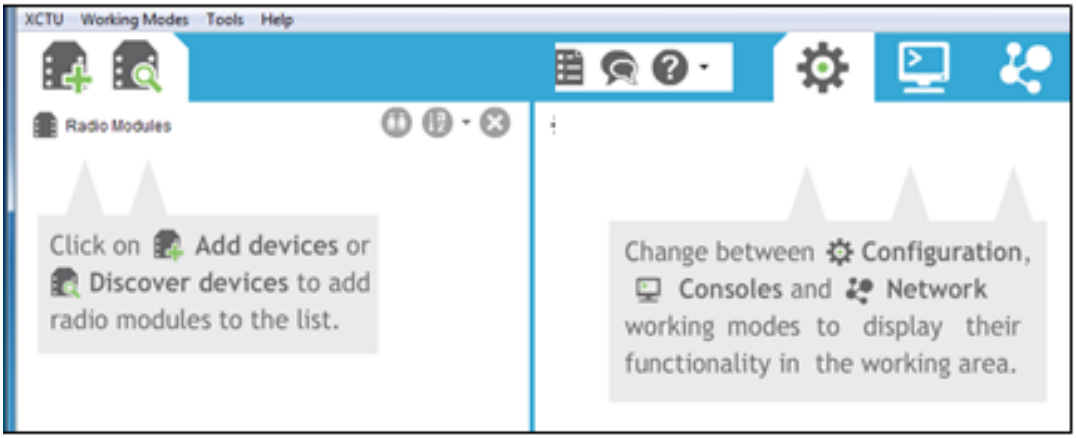


Figura 4.28: Ventana inicial XCTU
Fuente: Elaborado por el autor

Posteriormente aparece la pantalla que se muestra en la 4.29 se selecciona el puerto “COM” correspondiente en este caso el COM 10, y se presiona el botón siguiente.

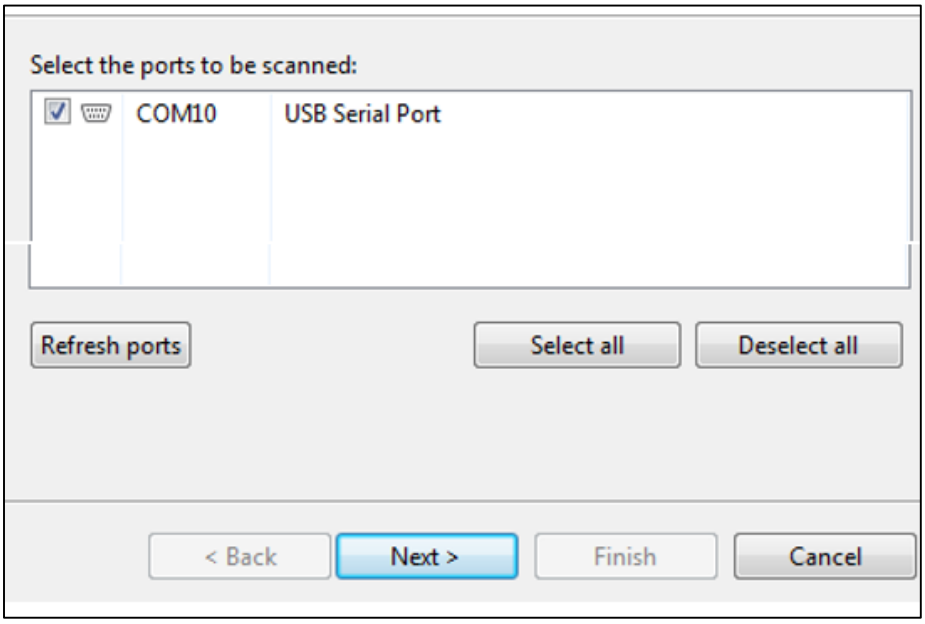


Figura 4.29: Selección puerto serial
Fuente: Elaborado por el autor

A continuación, aparece la pantalla que se muestra en la figura 4.30, se debe seleccionar la velocidad de transmisión, en este caso se selecciona 9600 porque la tarjeta Arduino trabaja con esta velocidad y se presiona en finalizar.

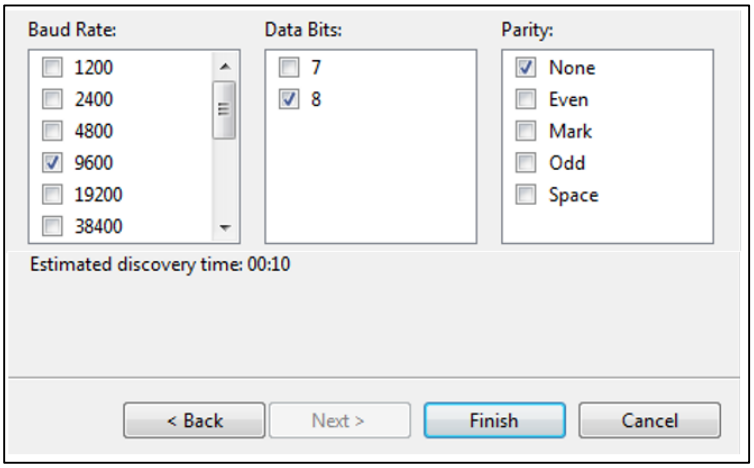


Figura 4.30: Selección velocidad de transmisión
Fuente: Elaborado por el autor

Sobre el modulo que se muestra en la figura 4.31, se realiza un clic para abrir los parámetros de configurar el módulo Xbee

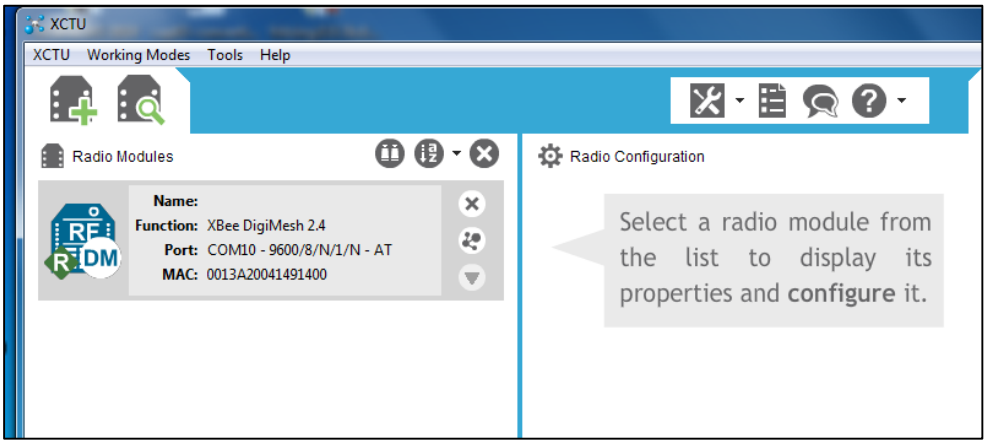


Figura 4.31: Ventana para la selección del módulo Xbee
Fuente: Elaborado por el autor

Luego aparece la pantalla que se muestra en la figura 4.32, y se realiza la configuración de los parámetros con fecha de color azul.

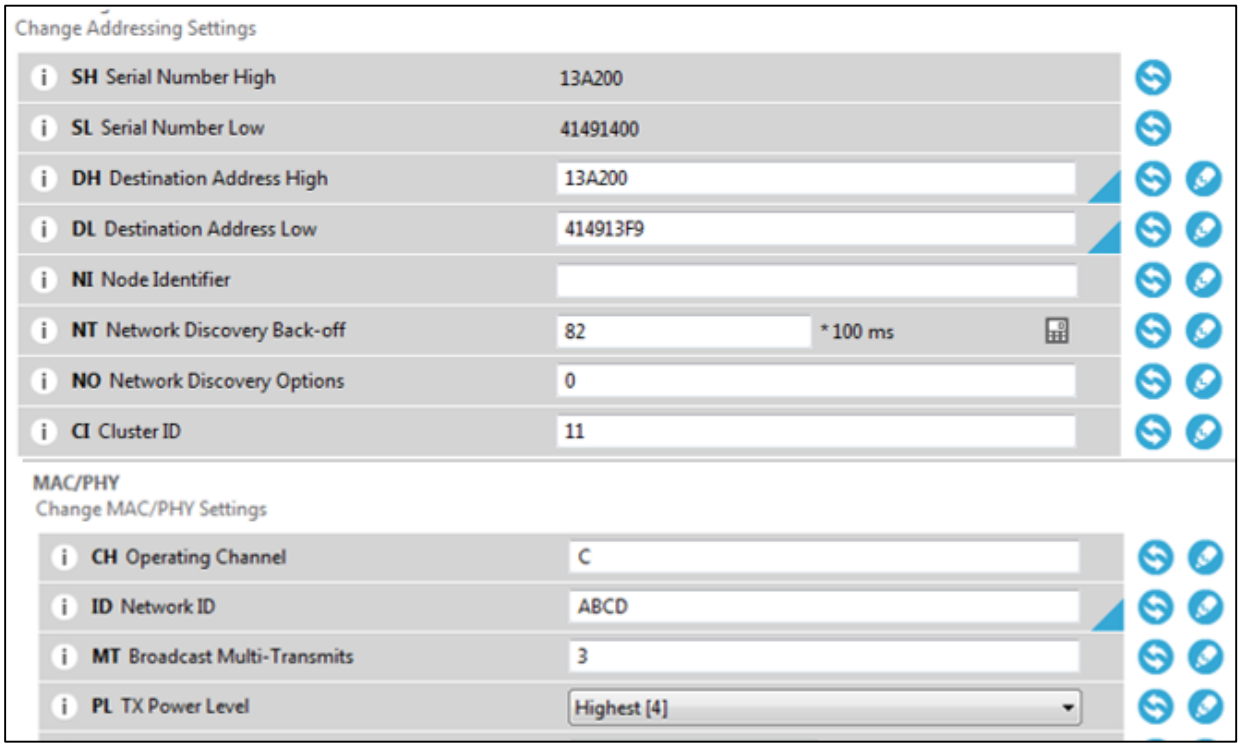


Figura 4.32: Parámetros de configuración modulo coordinador
Fuente: Elaborado por el autor

4.3.4 Configuración del nodo router

Los pasos para la configuración del nodo router son similares a los pasos del nodo coordinador. A continuación, en la figura 4.33 se muestra los parámetros que se han configurado.

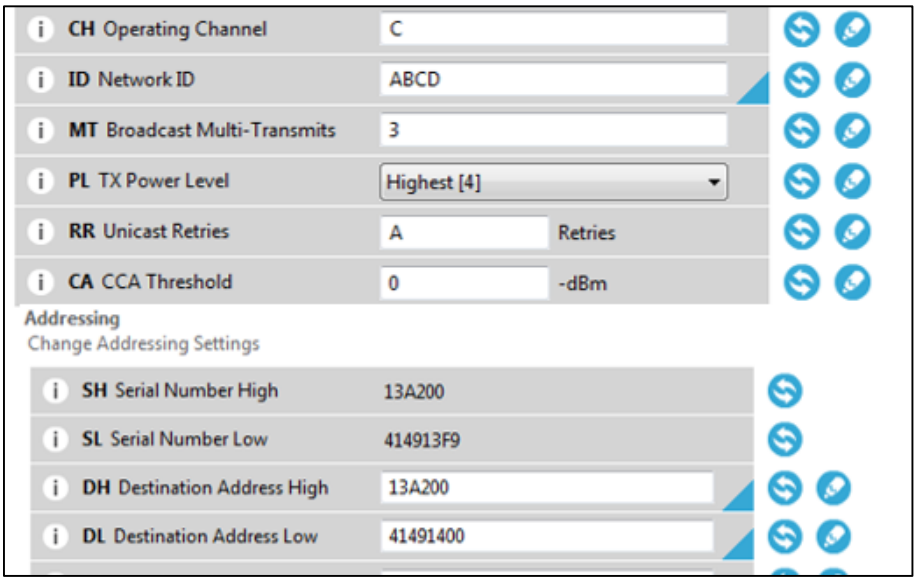


Figura 4.33: Parámetros de configuración módulo router
Fuente: Elaborado por el autor

4.4 Integración hardware y software

Para transferir el algoritmo se debe conectar el cable de transmisión de datos desde el computador hacia el puerto de comunicación del robot, como se muestra en la figura 4.34.

Figura 4.34: Transferencia de la programación
Fuente: Elaborado por el autor

4.5 Pruebas de funcionamiento

En esta parte se procede a realizar las pruebas para verificar el correcto funcionamiento del robot. Para esto se debe verificar el desplazamiento del robot en el modo manual y en el modo automático.

4.5.1 Posición de inicio

Esta rutina permite colocar los servomotores en una posición central. Como se observa en la figura 4.35 el robot se coloca correctamente en esta posición.

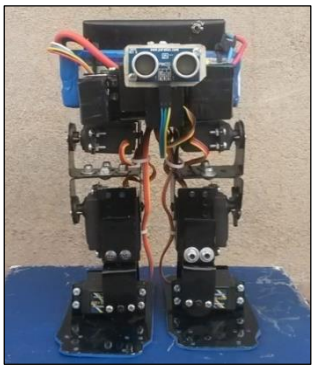


Figura 4.35: Posición central del robot
Fuente: Elaborado por el autor

4.5.2 Caminata hacia adelante

Esta rutina empieza al seleccionar en la interfaz de usuario el comando “adelante”. En la figura 4.36 se observa al robot desplazarse correctamente hacia adelante.

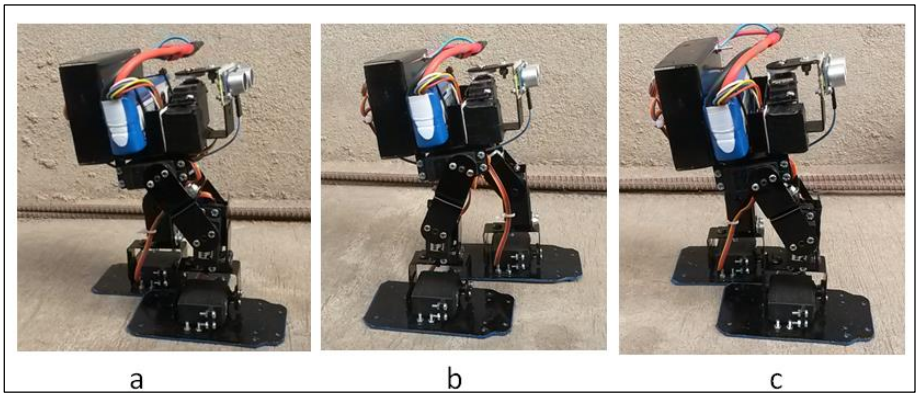


Figura 4.36: Caminata hacia adelante
Fuente: Elaborado por el autor

4.5.3 Giro a la izquierda

Esta rutina empieza al seleccionar en la interfaz de usuario el comando “izquierda”. En la figura 4.37 se muestra que el robot gira hacia la izquierda de forma correcta.

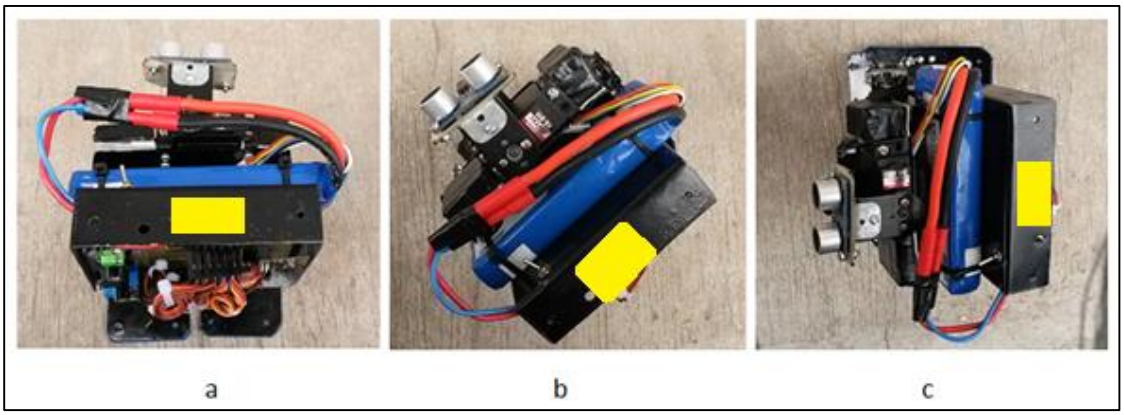


Figura 4.37: Giro a la izquierda
Fuente: Elaborado por el autor

4.5.4 Giro a la derecha

Esta rutina empieza al seleccionar en la interfaz de usuario el comando “derecha”. En la figura 4.38 se muestra que el robot gira hacia la derecha.

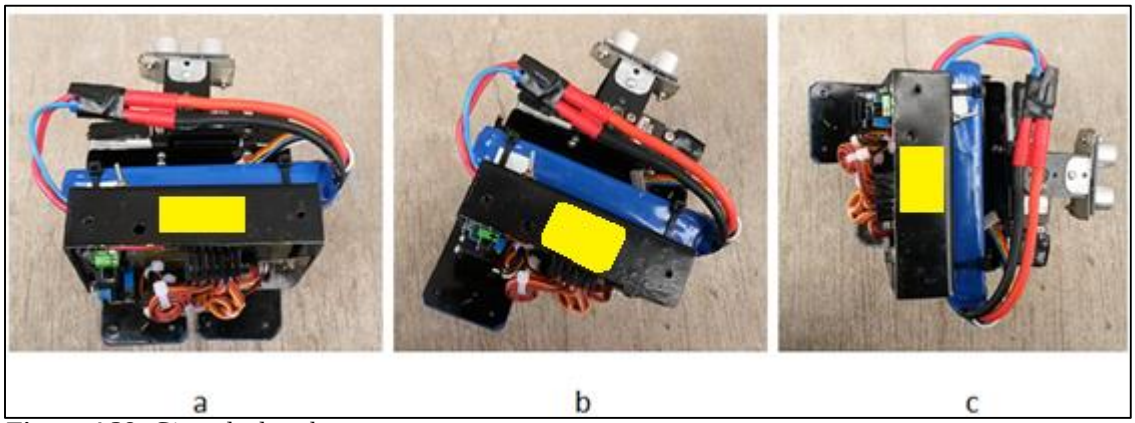


Figura 4.38: Giro a la derecha
Fuente: Elaborado por el autor

4.5.5 Rutina de control automático

Esta rutina empieza al seleccionar en la interfaz de usuario la opción “modo automático”. Esta opción permite al robot desplace de forma autónoma hacia adelante y también permite el funcionamiento del sensor. Como se observa en la figura 4.39, el robot gira para esquivar el obstáculo.

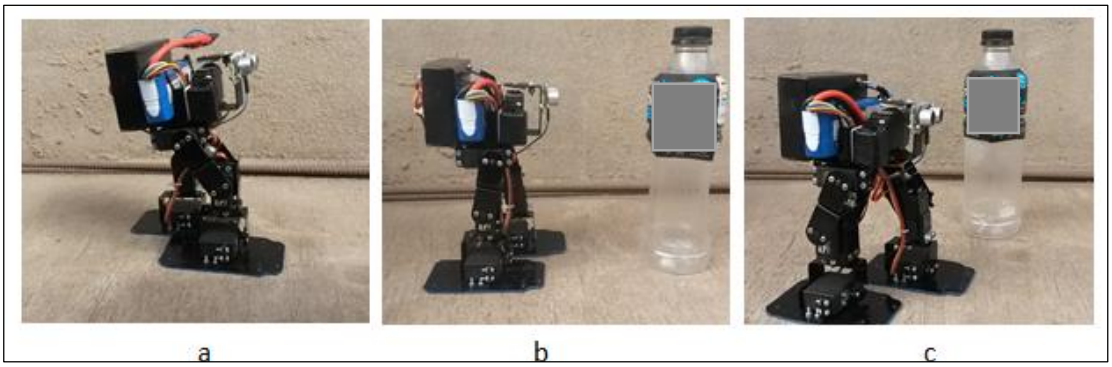


Figura 4.39: Evasión de obstáculos
Fuente: Elaborado por el autor

4.5.6 Autonomía de la batería

Luego de realizada las pruebas de funcionamiento del robot bípedo con los distintos movimientos de desplazamiento se determinó que el tiempo de uso de la batería es aproximadamente de 1hora y 30minutos. Esto ha sido determinado en una superficie plana de tipo sólida y sin ángulo de inclinación.

Tabla 4.2
Consumo de la batería

Desplazamiento del robot	Tiempo de duración de la batería
Continuo	90 minutos

Fuente: Elaborado por el autor

4.5.7 Pruebas de funcionamiento del sensor

Con referencia al sensor se procedió a comprobar el rango de detección del sensor para esto se programó distintos valores los cuales se muestra en la tabla 4.3 a continuación.

Tabla 4.3
Rango del sensor

Distancia(metros)	Detección
4.0 m	NO
3.5 m	NO
3.1 m	NO
3.0 m	NO
2.9 m	SI
1.0 m	SI
0.5 m	SI
0.2 m	SI
0.05 m	SI
0.03 m	SI
0.02 m	NO
0.01 m	NO

Fuente: Elaborado por el autor

4.5.8 Latencia entre el robot y la interfaz de usuario

En lo que se refiere a la distancia para la comunicación entre el robot bípedo y la interfaz de usuario se comprobó la latencia para esto se tomó como dato la distancia máxima para el enlace inalámbrico que es de 30 metros según los datos del fabricante de los módulos Xbee.

Tabla 4.4
Prueba de la comunicación

Comunicación con línea de vista	Funciona el enlace
1m	SI
2m	SI
10m	SI
15m	SI
18m	SI
22m	SI
25m	SI
29m	SI
30m	SI
31m	NO

Fuente: Elaborado por el autor

4.5.9 Medición de corriente en el circuito

Se procede a medir los valores de corriente y de voltaje para lo cual se utiliza un multímetro. Las mediciones obtenidas se observan en la figura 4.40.

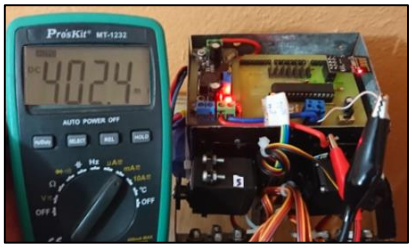


Figura 4.40: Corriente en el circuito
Fuente: Elaborado por el autor

En la tabla 4.5, se muestra los valores medidos de forma práctica tanto de la corriente como del voltaje del circuito de control. El valor de la potencia obtenida de forma práctica es de 2.02 Vatios (esta puede variar dependiendo el movimiento del robot), en cambio el valor obtenido de forma matemática es de 7.46 Vatios, por lo tanto, la potencia medida está dentro del rango de la potencia establecida mediante cálculos.

Tabla 4.5
Mediciones en el circuito de control

Corriente Total	Voltaje Total	Potencia Total
402.4 mA	5.02 V	2.02 W

Fuente: Elaborado por el autor

4.6 Análisis de resultados

Una vez concluido el presente proyecto se procedió a realizar el análisis de resultados, los cuales se realizan para verificar el correcto funcionamiento del producto funcional y que este a acorde a los objetivos planteados. Para esta revisión se ha tomado en cuenta el funcionamiento del sistema robótico tanto en el modo manual como en el modo automático.

En el modo de control manual el robótico es capaz de realizar las siguientes rutinas de movimientos: caminata hacia adelante, giro a la izquierda, giro a la derecha. En cambio, en modo automático el robótico es capaz de caminar hacia adelante y al detectar algún obstáculo predeterminado el robot procede a evadirlo.

En la tabla 4.6, se detalla la información resultante de las pruebas realizadas.

Tabla 4.6
Análisis de resultados

Ítem	Proceso	Cumple	No cumple	Observaciones
1	• Encendido del circuito	X		• Voltaje operación 5 voltios
2	• Conexión Xbee	X		• 30 metros con línea de vista
3	• Transmisión de datos robot e interfaz	X		• Se verifica mediante la interfaz
4	• Funcionamiento del sensor	X		• El sensor funciona con el modo automático
5	• Detección del obstáculo	X		• Detección dentro del rango de 25 cm
6	• Desplazamiento hacia adelante	X		• Mediante el botón adelante
7	• Desplazamiento hacia izquierda	X		• Mediante el botón izquierda
8	• Desplazamiento hacia la derecha	X		• Mediante botón derecha

Fuente: Elaborado por el autor

En la tabla 4.7 se muestra los resultados obtenidos en la prueba de la rutina hacia adelante, para esto se determinaron tres ítems, en los cuales se evalúan los resultados obtenidos. En esta prueba se determinó que, el sistema robótico trabaja con normalidad al desplazarse hacia adelante.

Tabla 4.7
Prueba rutina adelante

ÍTEM	**Prueba de la rutina de caminata hacia adelante**
Resultado	El robótico trabaja con normalidad al desplazarse hacia adelante
Error:	El robot pierde el equilibrio al desplazarse
Solución	Se calibra el movimiento de los servomotores

Fuente: Elaborado por el autor

En la tabla 4.8 se muestra los resultados obtenidos en la prueba de la rutina hacia la izquierda, para esto se determinaron tres ítems, en los cuales se evalúan los resultados obtenidos. En esta prueba se determinó que, el sistema robótico trabaja con normalidad al desplazarse hacia el lado izquierdo.

Tabla 4.8
Prueba rutina izquierda

ÍTEM	**Prueba de la rutina de giro hacia la izquierda:**
Resultado	Trabaja con normalidad al realizar el giro hacia el lado izquierdo.
Error:	Los movimientos del robot son imprecisos al girar
Solución	Se modifica el rango de giro del servomotor.

Fuente: Elaborado por el autor

En la tabla 4.9 se muestra los resultados obtenidos en la prueba de la rutina hacia la derecha, para esto se determinaron tres ítems, en los cuales se evalúan los resultados obtenidos. En esta prueba se determinó que, el sistema robótico trabaja con normalidad al desplazarse hacia el lado derecho.

Tabla 4.9
Prueba rutina derecha

ÍTEM	**Prueba de la rutina de giro hacia la derecha**
Resultado	El robótico trabaja con normalidad el giro hacia el lado derecho
Error:	Los movimientos del robot son imprecisos al girar
Solución	Se estable nuevos valores para el giro de los servomotores

Fuente: Elaborado por el autor

CONCLUSIONES

- Al analizar los parámetros de movilidad y estabilidad se determinó que el robot debe ser diseñado en base a los grados de libertad, el material de la estructura y los servomotores. Por lo cual al cumplir con estos parámetros se elaboró un prototipo de robot bípedo el cual es controlado de forma inalámbrica.

- Una vez analizado los componentes electrónicos, se seleccionó los elementos principales entre estos: el Arduino Mega, el sensor ultrasónico y los servomotores. Estos han sido seleccionados porque son compatibles con el software Arduino, por su consumo mínimo de corriente y por su tamaño reducido.

- Después de analizar los diferentes módulos de comunicación Xbee se determinó que el módulo S1 es el más adecuado, porque permite la transmisión de datos de forma instantánea, es óptimo para la comunicación punto a punto y por su bajo consumo de corriente. Este dispositivo se utilizó para controlar el robot de forma inalámbrica desde una interfaz de usuario.

- Una vez realizado el análisis entre diferentes softwares se ha decidido por Arduino, por qué es de fácil uso aun teniendo pocos conocimientos de programación y su entorno de desarrollo está basado en software libre. Por lo cual este software es adecuado para realizar el algoritmo de control del robot.

- Una vez realizado el análisis sobre los parámetros de construcción se determinó que el prototipo debe estar constituido por servomotores con un torque superior a 7 kg.cm, ya que esta fuerza es adecuada para conservar la estabilidad al colocarse de pie y porque son adecuados para mantener el desplazamiento de forma continua en una superficie plana que no tenga un ángulo de inclinación.

- Al realizar las pruebas de funcionamiento se verifico el algoritmo programado en el robot, para esto se utilizó la interfaz de usuario, en donde se comprobó la estabilidad, la movilidad y el control del robot.

RECOMENDACIONES

- Se recomienda que otro investigador podría agregarle visión artificial ya que de esta manera el robot podría ser utilizado para distinguir la forma y los colores de los obstáculos.

- Se recomienda agregar brazos, con los cuales el robot podrá realizar otros movimientos como sujetar y clasificar objetos.

- Es recomendable agregar una tarjeta de voz al circuito electrónico, esto permitirá construir un robot capaz de interactuar con las personas que se encuentran a su alrededor.

- Se recomienda realizar el estudio de la cinemática directa para imitar el movimiento bípedo mediante el software Matlab.

- Se recomienda realizar el estudio de la cinemática inversa para imitar el desplazamiento bípedo mediante el software Matlab.

- Se recomienda a otro investigador agregar un sistema de comunicación mediante una aplicación por Smartphone.

REFERENCIAS BIBLIOGRÁFICAS

Aprendiendo Arduino. (2017). Obtenido de Entender Arduino: https://aprendiendoarduino.wordpress.com/category/software/

Apuntes de aeromodelismo. (2017). Obtenido de Configurador de carga de baterías: http://aeromodelismo.epiel.com/scripts/carga.cgi?Lang=es&Tipo=LiPo&Voltaje=14.8&Capacidad=3000&CargaMaxima=&Modo=Lento&Cargador=SkyRC+iMAX+B6&.submit=Recalcular&.cgifields=Modo&.cgifields=Lang&.cgifields=Cargador&.cgifields=Tipo

Arduino.cl. (2018). Obtenido de http://arduino.cl/arduino-mega-2560/

Arias, F. (2006). *ebevidencia.com.* Obtenido de El proyecto de investigación: https://ebevidencia.com/wp-content/uploads/2014/12/EL-PROYECTO-DE-INVESTIGACI%C3%93N-6ta-Ed.-FIDIAS-G.-ARIAS.pdf

Asano, Y., Okada, K., & Inaba, M. (2017). *Design principles of a human mimetic humanoid: Humanoid platform to study human intelligence and internal body system.* Obtenido de http://robotics.sciencemag.org/content/2/13/eaaq0899

Autodesk. (2019). Obtenido de AutoCAD: https://latinoamerica.autodesk.com/products/autocad/overview

Balestrini, M. (2006). *issuu.* Obtenido de Como se elabora el proyecto d investigacion : https://issuu.com/sonia_duarte/docs/como-se-elabora-el-proyecto-de-inve

Bonell, M. (2011). *DISEÑO Y CONSTRUCCIÓN DE UN ROBOT HUMANOIDE.* Barcelona.

Boston Dynamics. (s.f.). Obtenido de Spot: https://www.bostondynamics.com/spot

Cavasassi, J. (2019). *Cavadevices.* Obtenido de Cómo calcular la capacidad necesaria de batería para almacenar energía: http://www.cavadevices.com/archivos/FOLLETOS/calculo%20de%20bateria.pdf

Cela, A. (2012). *Teleoperación inalámbrica de robot humanoide mediante traje con sensores de tipo resistivo lineal.* Madrid.

Cevallos, M. (2019). *Universidad Israel.* Obtenido de http://157.100.241.244/handle/47000/1936

Chacin, M., & Roseli, S. (1999). *mindspages.* Obtenido de Desarrollo de un sistema robotico para eludir obstaculos: http://www.mindspages.net/files/tesis.pdf

Cruz, J. (2017). *Aplicación de control de movimientos y recopilación* . Sevilla.

Díaz , V. (2013). *Prezi.* Obtenido de Método de Modelación: https://prezi.com/tjwhdiid8yjq/metodo-de-modelacion/

DIGI. (2019). Obtenido de XCTU : https://www.digi.com/products/iot-platform/xctu

Durán, E., & Granja, D. (2010). *Diseño y construcción de un kit didáctico de experimentación.* Obtenido de Universidad Politécnica Salesiana: https://documentop.com/diseo-y-construccion-de-un-kit-didactico-de-experimentacion-_59866f891723ddb404628a34.html

Escalona, M., & Morillo, J. (2017). *Teoría Clásica de Control Automatico y en Ingeniería.* Quito: Jurídica del Ecuador.

Flickr. (2010). Obtenido de Asimo: https://www.flickr.com/photos/arselectronica/4852180795

García, A. (2013). *Panama Hitek.* Obtenido de Arduino Mega: Características, Capacidades y donde conseguirlo en Panamá: http://panamahitek.com/arduino-mega-caracteristicas-capacidades-y-donde-conseguirlo-en-panama/

GoTronic. (s.f.). Obtenido de Châssis WILD THUMPER: https://www.gotronic.fr/art-chassis-wild-thumper-12393.htm

Granja, M. (2014). *Escuela Politécnica Nacional.* Obtenido de https://bibdigital.epn.edu.ec/browse?type=author&value=Granja+Ram%C3%ADrez%2C+Mario+German

Guzmán , C. (2010). *Construcción de un Robot Bípedo Basado en Caminado Dinámico.* Cuernavaca.

Herrera, M. (2009). *Ensamblaje y control de una plataforma robótica bípeda mediante un PC.* Obtenido de Escuela Politécnica Nacional: http://bibdigital.epn.edu.ec/bitstream/15000/1917/1/CD-2471.pdf

Hizzook. (28 de Octubre de 2008). Obtenido de Troody the Robotic Dinosaur: http://www.hizook.com/blog/2008/10/28/troody-robotic-dinosaur

Hobbyking - Servo. (s.f.). Obtenido de Power HD: https://cdn-global-hk.hobbyking.com/media/file/0/6/068000024-0_-125352_-storm-5_-spec_2_.pdf

Hobbyking. (2019). Obtenido de B6 iMAX: https://cdn-global-hk.hobbyking.com/media/file/506780627X258206X26.pdf

Hubor. (2015). Obtenido de Cómo es el trabajo con Proteus?: http://www.hubor-proteus.com/proteus-pcb/proteus-pcb/230-como_trabajar_con_proteus.html

Intelligence Artificielle. (s.f.). Obtenido de Les robots marcheurs de Honda: http://intelligence-artificielle-tpe.e-monsite.com/album-photos/des-exemples-bluffants/asimo-evolution.html

Lezama, D., & Sklar, A. (2004). *Construcción de Robots Bípedos.* Uruguay.

Lozada, J. (2014). *Dialnet.* Obtenido de Investigación Aplicada: https://dialnet.unirioja.es/servlet/articulo?codigo=6163749

Lynxmotion. (2018). Obtenido de Robot Bípedo Brat : http://www.lynxmotion.com/c-139-auton-combo-kit.aspx

Martínez, D. (2018). *Repositorio Universidad de Guayaquil.* Obtenido de Robonoide bípedo autónomo realizado en plataforma Arduino.: http://repositorio.ug.edu.ec/handle/redug/36172

Méndez, J. (2018). *Repositorio uisrael.* Obtenido de Desarrollar una válvula de riego agricola motorizda programable mediante Bluetooth, autónoma y movil: https://repositorio.uisrael.edu.ec/handle/47000/1664

Microsoft . (2019). Obtenido de Visual Studio: https://docs.microsoft.com/es-es/visualstudio/get-started/visual-studio-ide?view=vs-2019

Morillo, E., & Portero, J. (2014). *Diseño y construcción de un prototipo de robot bípedo autónomo con desplazamiento dinámico para el laboratorio de robótica industrial de la ESPE de Latacunga.* Obtenido de Escuela Superior Politécnica del Ejército: https://docplayer.es/9195796-Departamento-de-ciencias-de-la-energia-y-mecanica-carrera-de-ingenieria-mecatronica.html

Naylamp. (2018). *Naylamp Mechatronics*. Obtenido de Convertidor Voltaje DC-DC : https://naylampmechatronics.com/conversores-dc-dc/196-convertidor-voltaje-dc-dc-step-down-3a-lm2596.html

Onsemi. (2019). Obtenido de https://www.onsemi.com/pub/Collateral/LM2596-D.PDF

Ortega, D. (2012). *Scribd.* Obtenido de revisión documental: https://es.scribd.com/document/89555504/Revision-Documental

Oyarce, A. (2010). *Scribd.* Obtenido de Guía de usuario XBEE Serie 1: https://es.scribd.com/document/136968795/XBee-Guia-Usuario

Parallax. (2018). Obtenido de PING))) Ultrasonic Distance Sensor: https://www.parallax.com/product/28015

Parra, F. (2015). *Sistemas Digitales, Computación 4.* Obtenido de Instituto Tecnológico de Mérida Área Técnica: : http://aryan.church/5916989-Instituto-tecnologico-de-merida.html

Plataformas Zigbee. (2014). Obtenido de Plataformas Zigbee: http://plataformaszigbee.blogspot.com./2012/11/wsn-zigbee-vs-wifi.html

PUCP. (2016). Obtenido de Blog PUCP: http://blog.pucp.edu.pe/blog/cristhianjc/wp-content/uploads/sites/791/2015/10/Manual.pdf

Ramírez, R., & Reyes, R. (2015). *Diseño e implementacion de un robot autónomo móvil usando tecnología FPGA.* Guayaquil.

Rancel, M. (2019). *APR.* Obtenido de https://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

Robot Shop. (2019). Obtenido de https://www.robotshop.com/media/files/pdf/arduinomega2560datasheet.pdf

RobotShop . (2018). Obtenido de Parallax PING Ultrasonic Sensor: https://www.robotshop.com/en/parallax-ping-ultrasonic-sensor.html

Rodríguez, F., & Berenguel, M. (2012). *Grupo de investigación: Automática, robótica y mecatrónica*. Obtenido de ARM: http://aer.ual.es/docencia_es/sr/docencia/tema2_robotica%20manipulacion.pdf

Ruiz , J. (2012). *Academia.* Obtenido de Arduino + XBee: https://www.academia.edu/16412009/Arduino_XBee

T-BEM. (2018). Obtenido de Baterías LiPo: http://blog.teslabem.com/como-usar-y-cuidar-las-baterias-lipo/

Tdrobotica. (2018). Obtenido de XBee Explorer USB: http://tdrobotica.co/xbee-explorer-usb/651.html

Tecnologia. (2016). Obtenido de http://tecnologiagema2000seda.blogspot.com/2016/04/tipos-de-sistemas-de-control.html

Thayer, L. (2017). *MCI Electronic*. Obtenido de Arduino UNO: http://arduino.cl/arduino-uno/

Wired. (2004). Obtenido de The Humanoid Race: https://www.wired.com/2004/07/race/

ANEXO 1

MANUAL DE USUARIO

MANUAL DE USUARIO

IMPLEMENTACION DE UN PROTOTIPO DE ROBOT BÍPEDO CON TECNOLOGÍA XBEE

INTRODUCCIÓN

El manual de usuario es una guía para el correcto uso de las partes del robot bípedo y para explicar el modo de uso de la interfaz de usuario.

En esta guía se presenta la forma física del robot, los elementos que lo componen, la forma de activar el funcionamiento, la forma de transferir la programación y como cargar correctamente la batería. De igual manera se explica cómo instalar el software para utilizar la interfaz de usuario.

1. PREVENCIÓN DE LESIONES Y DAÑOS


Peligro: Este símbolo se utiliza para indicar peligro mortal o lesiones graves, si no se respetan las instrucciones.


Advertencia: Se utiliza cuando puede haber lesiones leves o daños al equipo electrónico, si no se siguen las instrucciones.


Precaución: Se utiliza cuando puede haber lesiones leves o daños al equipo electrónico.

1.1 Peligro

- Se debe respetar las normas de seguridad cuando el robot bípedo esté en funcionamiento.
- El buen funcionamiento del robot bípedo no está garantizado, ya que se trata de un prototipo, que puede funcionar mal debido al modelado de la programación.
- Utilice el robot en lugares con ventilación suficiente para evita recalentamiento en las partes electrónicas.

1.2 Advertencia

- Mantenga el robot lejos de los niños. Aunque parece un juguete este podría herir a un niño si no está bajo observación de un adulto.

- Si la batería se rompe apáguela y desconéctela inmediatamente, ya que, al ser expuesta a la humedad, líquidos o llamas la persona puede sufrir un accidente.

- Nunca desmonte o manipule la circuitería de los servomotores.

- No utilice el robot en condiciones calurosas, humedad o frio. El prototipo está formado por componentes electrónicos puede producirse errores.

- Compruebe que el conector de carga esté bien apretado. Quítelo en cuanto haya terminado el proceso de carga.

- Por favor, lea atentamente el manual para evitar cualquier percance.

1.3 Precaución

- Alargue el tiempo de vida útil del robot si lo usa en una superficie plana y lisa. Si la superficie es irregular el robot se podría caer y estropearse.

- Nunca sostenga el robot bípedo durante su funcionamiento.

- Si utiliza el robot, fuera de un lugar tenga cuidado de no derramar algún liquido sobre el circuito de control ya que este podría explotar.

2. ESTRUCTURA FISICA

La estructura del robot está elaborada a base del tol galvanizado. Este material se caracteriza por ser resistente y por ser liviano.

2.1 Vista frontal y lateral

En la figura 1 se muestra la vista frontal y lateral del robot bípedo.

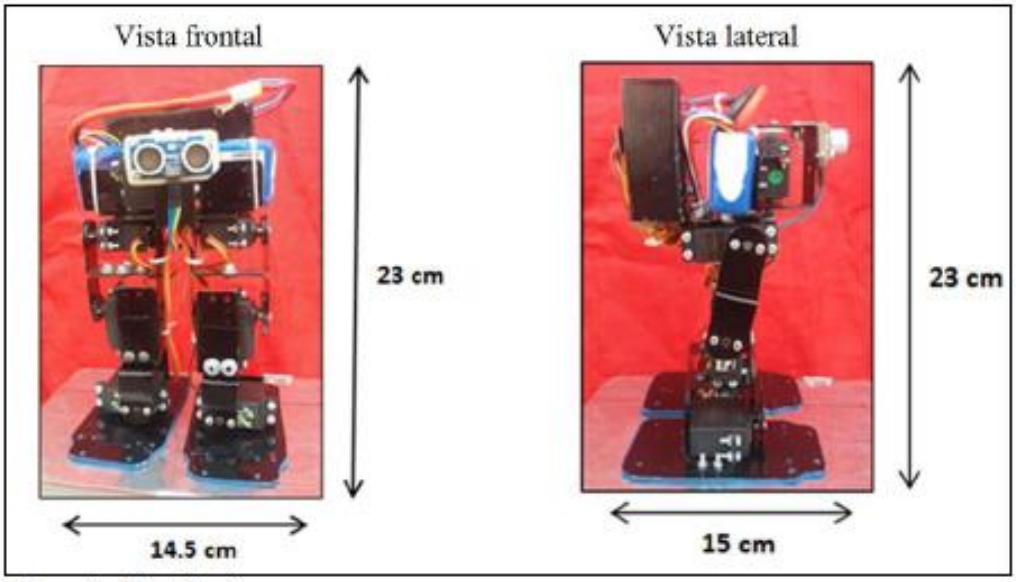


Figura 1: Robot bípedo
Fuente: El autor

2.2 Componentes del robot

El robot bípedo está compuesto por los siguientes componentes que se muestran a continuación:

1) Sensor de detección
1) Batería de litio
1) Circuito electrónico.
1) Arduino Mega
1) Estructura ensamblada

El robot está formado por siete grados de libertad a continuación en la figura 2 se muestra los componentes y los grados de libertad.

- Primer grado de libertad en el Tobillo izquierdo (1 GDL)
- Segundo grado de libertad en el Tobillo derecho (2 GDL)
- Tercer grado de libertad en la rodilla izquierda (3 GDL)
- Cuarto grado de libertad en la rodilla derecha (4 GDL)
- Quinto grado de libertad en la pierna izquierda (5 GDL)
- Sexto grado de libertad en la pierna derecha (6 GDL)
- Séptimo Grado de libertad en la cabeza (7 GDL)

Figura 2: Partes del robot
Fuente. El autor

2.3 Switch de encendido / apagado

El *switch* está ubicado en la parte superior, su función es encender todo el sistema del robot bípedo y se muestra en la figura 3.

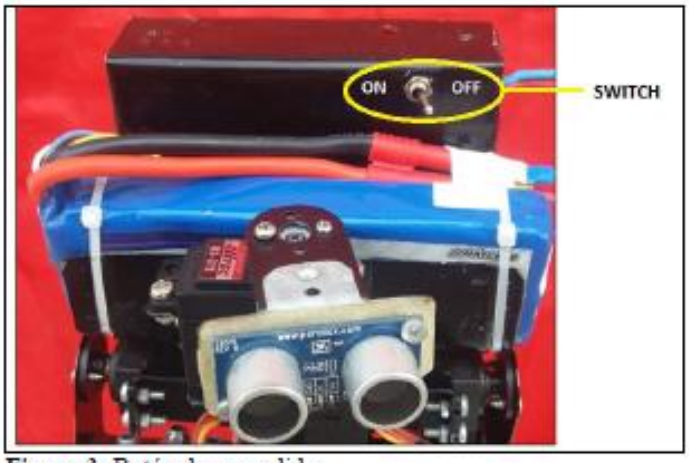


Figura 3: Botón de encendido
Fuente: El autor

2.4 Conector USB

El conector USB que se muestra en la figura 4, se utiliza para transferir el algoritmo de programación hacia el Arduino Mega, mediante el cable de conexión.

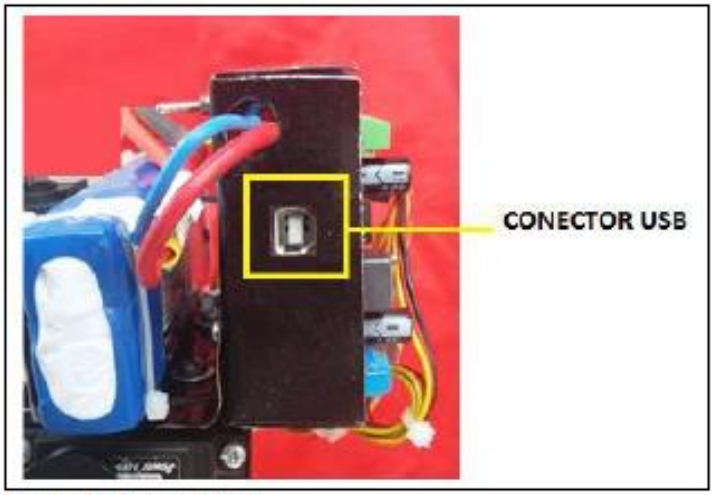


Figura 4: Conector USB
Fuente: El autor

2.5 Conector para cargar la batería

La carga de la batería se realiza mediante el conector de cinco pines que se muestra en la figura 5. Este cable permite cargar de forma balanceada las celdas de la batería, equilibrando el voltaje de forma correcta el voltaje para evitar daños en la batería.

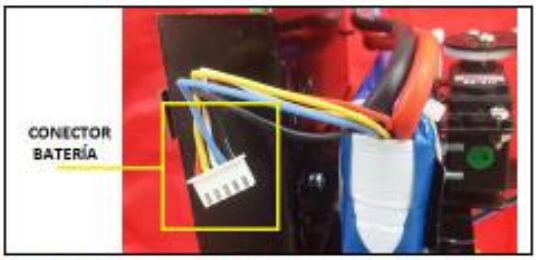


Figura 5: Conector de la batería
Fuente: El autor

En la figura 6 se muestra la forma de configurar el cargador de baterías iMAX B6 para que cargué de forma balanceada las 4 celdas de la batería.

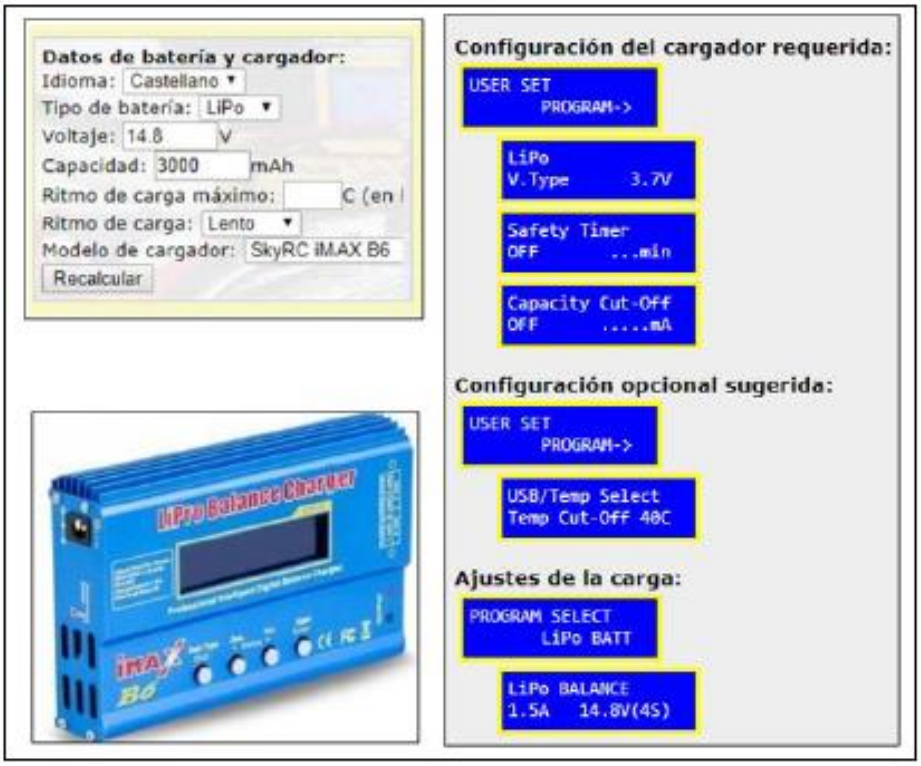


Figura 6: Conexión del cargador
Fuente: (Apuntes de aeromodelismo, 2017)

3. INSTALACIÓN DEL SOFTWARE VISUAL STUDIO

a) Descargar el programa en el link https://visualstudio.microsoft.com/es/downloads/. La versión que se ha utilizado para este proyecto es la 2017, pero se puede descargar una igual o superior ya que funciona correctamente.

b) Se ejecuta el instalador para visualizar el asistente de instalación, como se muestra en la figura 7.

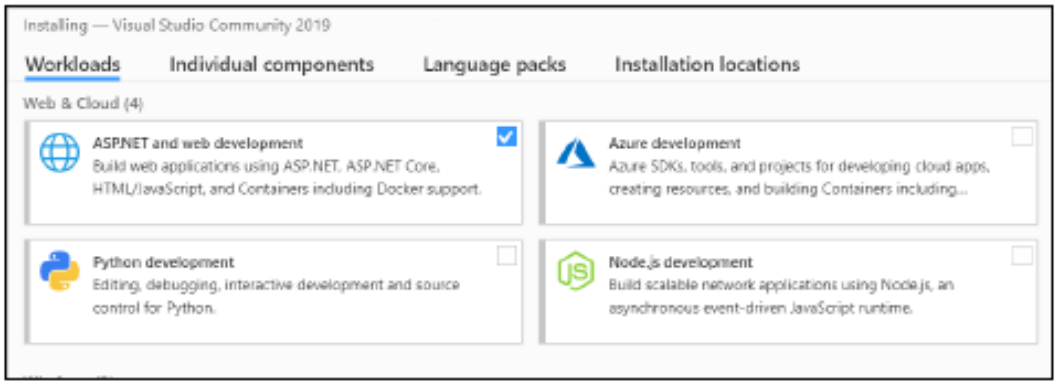


Figura 7: Asistente de instalación
Fuente: (Microsoft, 2019)

c) Seleccionar los componentes a utilizar, en este caso se requiere de *Visual Basic*, como se observa en la figura 8.

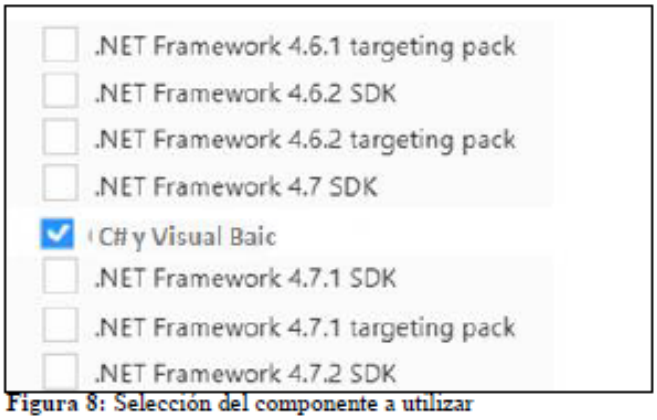


Figura 8: Selección del componente a utilizar
Fuente: (Microsoft, 2019)

d) A continuación, se elige la opción instalar, como se muestra en la figura 9.

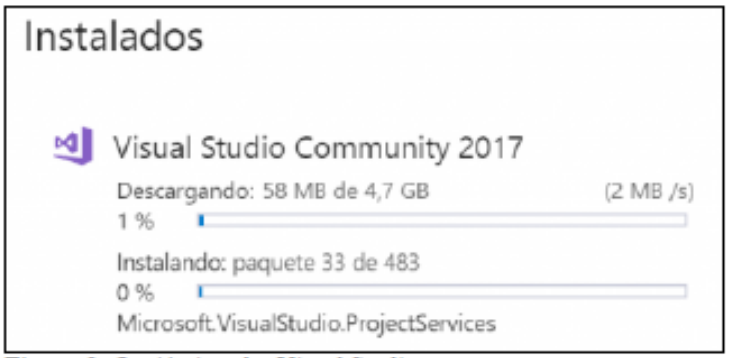


Figura 9: Opción instalar Visual Studio
Fuente: (Microsoft, 2019)

e) Una vez instalado se debe reiniciar el equipo. Posteriormente se Después elegir la opción, *De momento no; quizás más tarde*. Como se muestra en la figura 10.

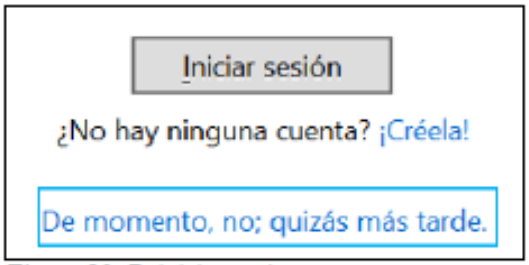


Figura 10: Reiniciar equipo
Fuente: (Microsoft, 2019)

f) El último paso es pulsar en la opción *iniciar Visual Studio*. (figura 11).

Figura 11: Iniciación Visual Studio
Fuente: (Microsoft, 2019)

4. INSTALACIÓN DEL SOFTWARE XCTU

a) Descargar el programa en el link https: https://www.digi.com/products/embedded-systems/digi-xbee-tools/xctu . La versión que se ha utilizado en este proyecto es la 2019 por lo cual se debe descargar una versión igual para que funcione correctamente.

b) Ejecuta el instalador y seleccionar la opción *Next*, como se muestra en la figura 12.

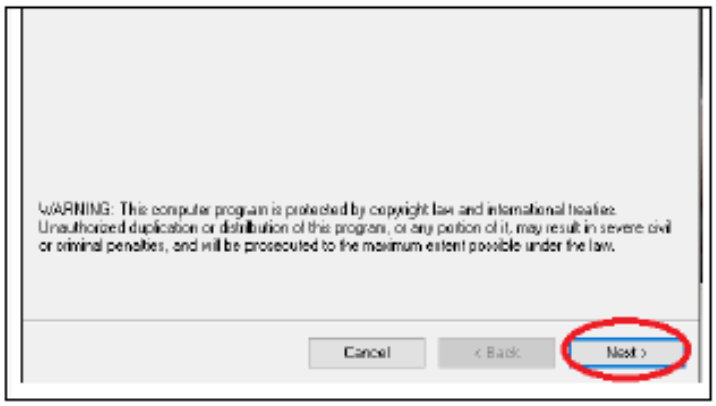


Figura 12: Asistente de instalación
Fuente: (DIGI, 2019)

a) Aceptar los términos y dar clic sobre la opción *Next*, como se muestra en la figura 13.

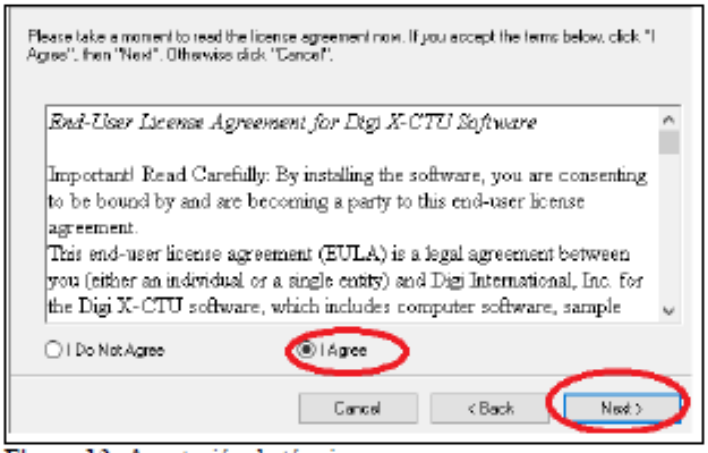


Figura 13: Aceptación de términos
Fuente: (DIGI, 2019)

b) A continuación, seleccionar la opción *Everyone* y dar clic sobre la opción *Next*, como se muestra en la figura 14.

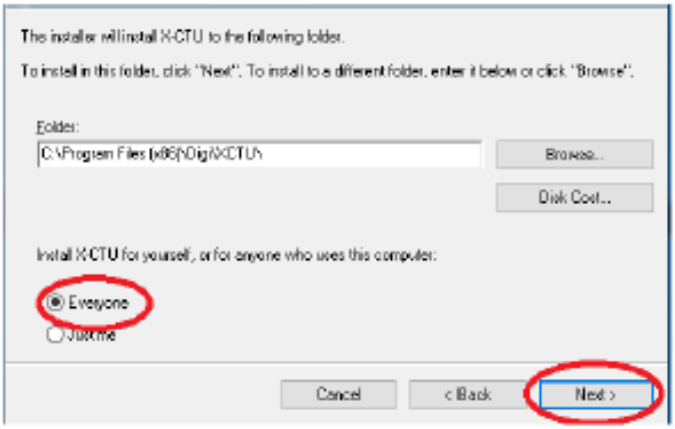


Figura 14: Selección Everyone
Fuente: (DIGI, 2019)

c) El programa comienza a instalarse, como se muestra en la figura 15.

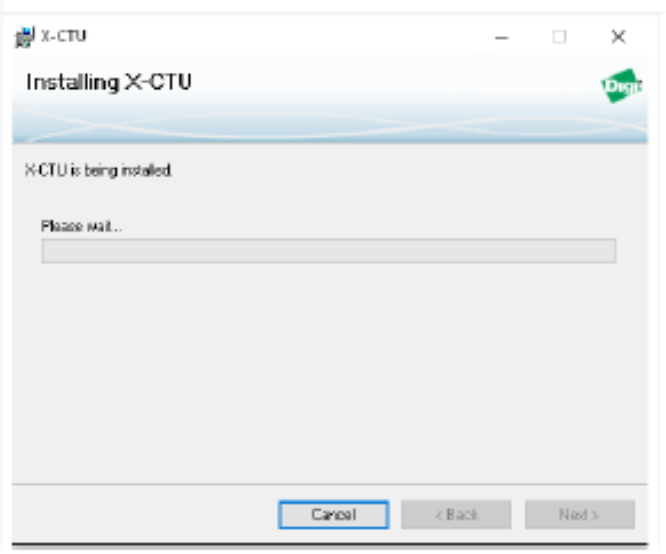


Figura 15: Instalación XCTU
Fuente: (DIGI, 2019)

5. INTERFAZ DE USUARIO

La interfaz de usuario proporcionar un entorno visual sencillo que ha sido desarrollado para controla de forma manual y de forma automática el desplazamiento del robot. Para iniciar este programa se debe dar clic sobre la opción *Iniciar*, como se muestra en la figura 16.

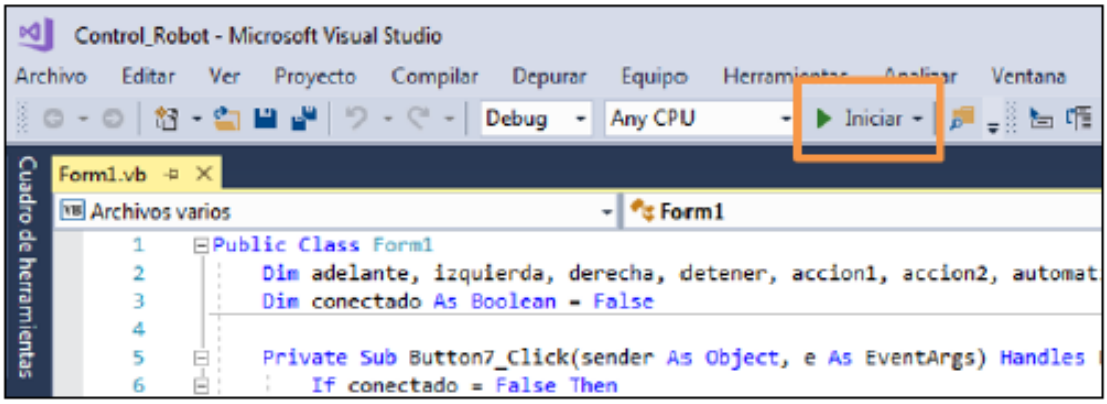


Figura 16: Botón Iniciar
Fuente: (Microsoft, 2019)

5.1 Configuración

Seleccionar la opción configuración y establecer el puerto *COM* designado por el computador, en este caso es el *COM 10*, como se observa en la figura 17.

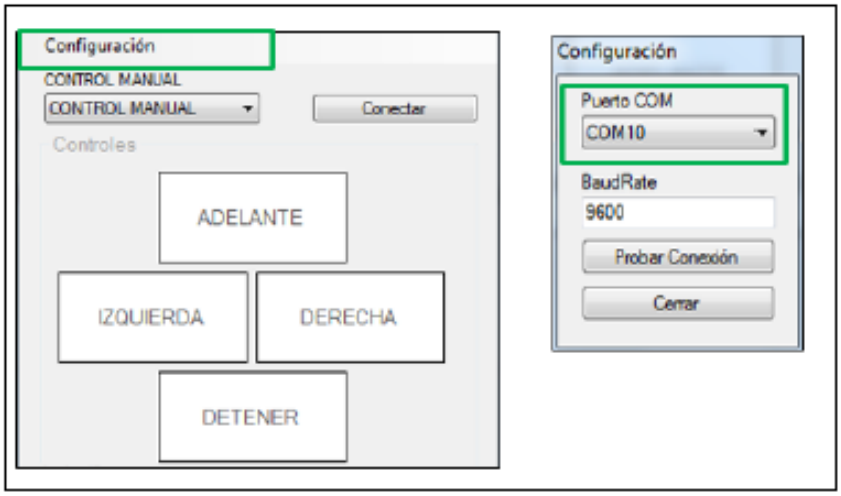


Figura 17: Configuración de puerto *COM*
Fuente: (Microsoft, 2019)

5.2 CONTROL MANUAL

El funcionamiento manual empieza al seleccionar la opción *Control Manual,* como se observa en la figura 18. El control manual permite dirigir el desplazamiento del robot mediante los botones de dirección.

Figura 18: Botones modo manual
Fuente: (Microsoft, 2019)

A continuación, se muestra en la tabla 1 el funcionamiento de los botones que se encuentran activados en la función *Control Manual.*

Tabla 1
Botones del control manual

Botón	Función
CONTROL MANUAL	• Activa el control manual del robot.
IZQUIERDA	• Permite el movimiento del robot hacia la izquierda.
DERECHA	• Permite el movimiento del robot hacia la derecha.
DETENER	• Detiene el desplazamiento del robot.

Fuente: El autor

5.3 CONTROL AUTOMÁTICO

El funcionamiento automático empieza al seleccionar la opción *Control Automático,* como se observa en la figura 19. Esta función le permite al robot desplazarse de forma autónoma y esquivar obstáculos predeterminados, los cuales deberán estar dentro del rango de detección.

Figura 19: Interfaz modo automático
Fuente: (Microsoft, 2019)

A continuación, en la tabla 2 se muestra el funcionamiento de los botones que se activan en la función *Control Automático*.

Tabla 2
Botones Control Automático

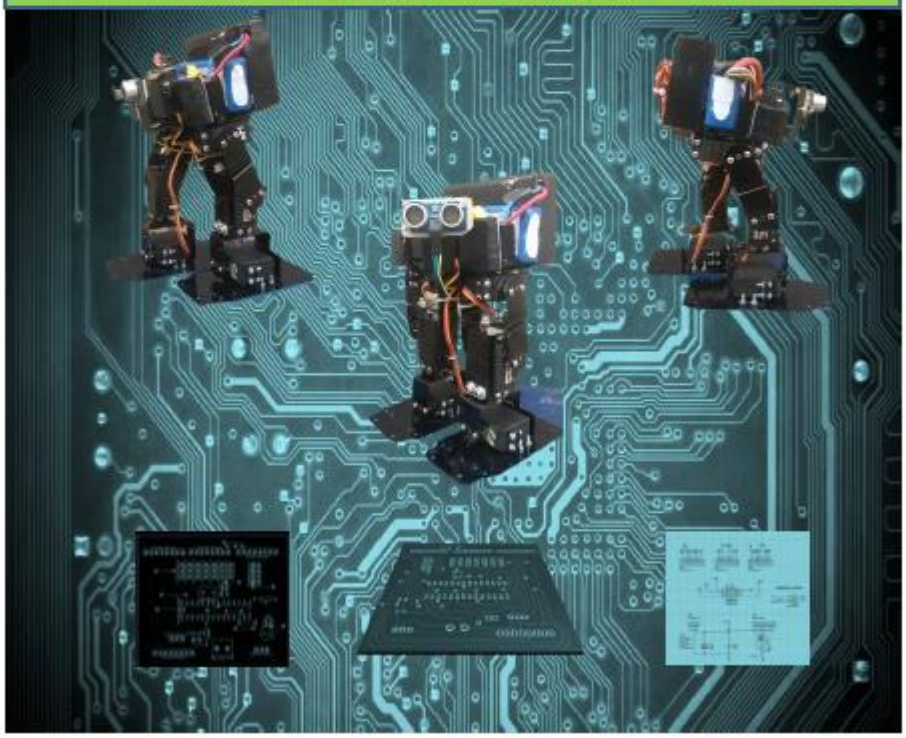
Botón	Función
CONTROL AUTOMATICO	Activa el desplazamiento automático del robot.
Sensor de Evasión	Indica la detección de un obstáculo.

ANEXO 2

MANUAL TÉCNICO

MANUAL TÉCNICO

ROBOT BÍPEDO CON TECNOLOGÍA XBEE PARA LA DETECCIÓN DE OBSTÁCULOS



INTRODUCCIÓN

El manual técnico es una guía para un público con conocimientos técnicos básicos de electrónica y ha sido elaborada para identificar los aspectos y características de funcionamiento del robot bípedo. En esta documentación se explica los diagramas electrónicos del circuito de control y las características de los elementos que la constituyen.

1. COMPONENTES PRINCIPALES DEL ROBOT

El robot está compuesto por 7 grados de libertad, un sensor de distancia, una batería de litio, y por el circuito electrónico. En la figura 1 se muestra las partes del robot.

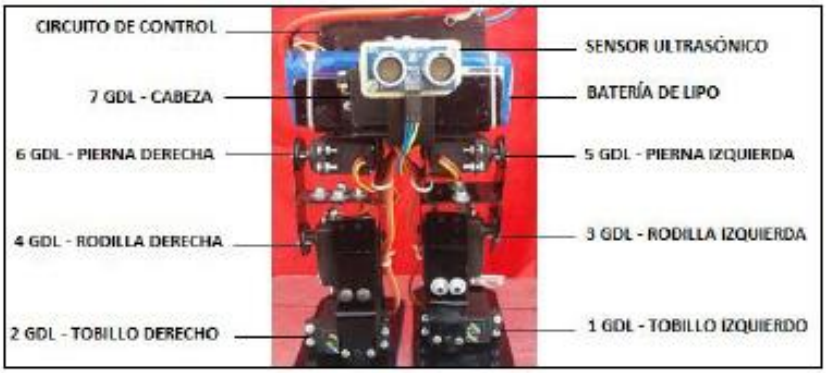


Figura 1: Partes del robot
Fuente: El autor

1.1 Arduino Mega

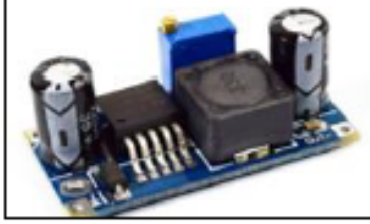
El Arduino Mega es una tarjeta de desarrollo que trabaja con código abierto, está constituido por un microcontrolador Atmega2560 y es muy utilizado para el desarrollo de proyectos de robótica. En la figura 2 se muestra las características técnicas de esta tarjeta.

Parámetro	Descripción
Microcontrolador	ATmega2560
Voltaje de entrada	7-12 Voltios (operativo 5 Voltios)
Pines digitales	54 pines digitales (entada/salida)
Pines análogas	16 pines (entradas)
Memoria flash	256KB
Corriente pines	40 mA (miliamperios)

Figura 2: Características Arduino mega
Fuente: El autor

1.2 Convertidor de voltaje LM2596

Su función es transformar un nivel de voltaje a otro de menor nivel. Para un buen funcionamiento el voltaje de entrada debe ser superior al voltaje de salida por lo menos en 1.5 voltios (V). En la figura 3 se muestra las características técnicas.

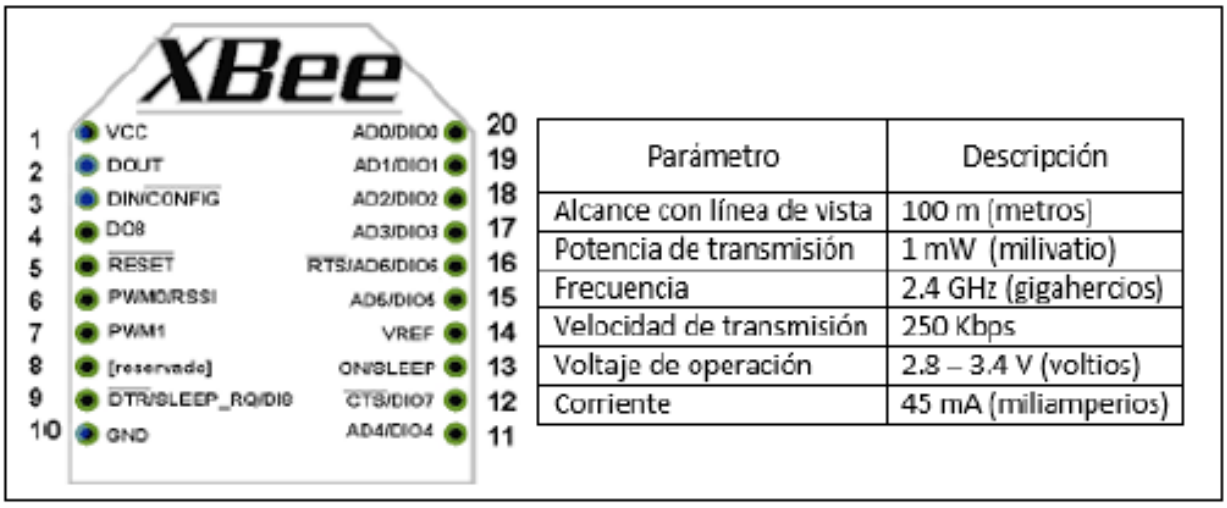


LM2596	
Voltaje de entrada	4.5V a 40V
Voltaje de salida	1.23V a 37 V
Corriente salida Máxima	3A

Figura 3: Características del LM2596
Fuente: El autor

1.3 Modulo Xbee

Es fabricado por Digi International y es utilizado para dar soluciones de conexión inalámbrica. El módulo Xbee que se utiliza pertenece a la serie 1, el cual es utilizado en redes de trabajo punto a punto y punto a multipunto. Estos módulos son económicos y fáciles de utilizar, entre sus características destacan su bajo consumo de energía y su largo alcance con línea de vista. En la figura 4 se muestra las características técnicas.



Parámetro	Descripción
Alcance con línea de vista	100 m (metros)
Potencia de transmisión	1 mW (milivatio)
Frecuencia	2.4 GHz (gigahercios)
Velocidad de transmisión	250 Kbps
Voltaje de operación	2.8 – 3.4 V (voltios)
Corriente	45 mA (miliamperios)

Figura 4: Características del LM2596
Fuente: El autor

1.4 Servomotor

Este componente permite el movimiento de las articulaciones del robot. Está formado por un motor de corriente continua y por un sistema de engranajes para el control de posición. Para su funcionamiento cuenta con tres cables. El cable rojo para la alimentación positiva, el cable negro para la conexión a tierra y el último cable para el control de posición. La estructura del robot bípedo está formada por tres tipos de servomotores y se muestran en la tabla 1.

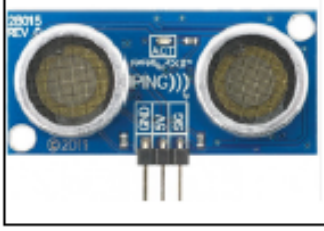
Tabla 1
Características servomotores

CARACTERISTICA	TURNIGY-1501	POWER-HD	HITEC-311
Voltaje de operación	4.8 a 6.0 VDC	4.8 a 6.0 VDC	4.8 a 6.0 VDC
Engranaje tipo	Metal	Metal	Plástico
Torque (4.8 - 6Vdc)	15.5 – 17 kg-cm	8.6 kg./cm	3.53 kg./cm
Velocidad	0.16 - 0.14 sec	0.17 - 0.14 sec	0.15 - 0.14 sec
Modificable a 360°	Si	Si	Si
Dimensiones	40.7x20.5x 39.5 mm	41.9 x 20.6 x 41.9	39.9 x 19.8 x 36.6
Peso	60.0 g	56.0 g	56.0 g

Fuente: El autor

1.5 Sensor Ultrasónico

El sensor está diseñado para medir la distancia de los objetos en un rango de 3 centímetros (cm) hasta los 3 metros (m). Está constituido por tres terminales, el pin GND como referencia a tierra, el pin 5V como alimentación positiva y el pin SIG como entrada y salida de datos. En la figura 5 se muestra las características técnicas.



Parámetro	Descripción
Dimensiones	22 x 46 x 16 mm
Rango de medición	3 cm – 3 m
Voltaje de alimentación	5 V
Consumo de corriente	30 - 35 mA
Frecuencia de ultrasonido	40kHz

Figura 5: Características sensor ultrasónico
Fuente: El autor

2. DIAGRAMA ESQUEMÁTICO

En la figura 6 se muestra el diagrama esquemático de la parte electrónica.

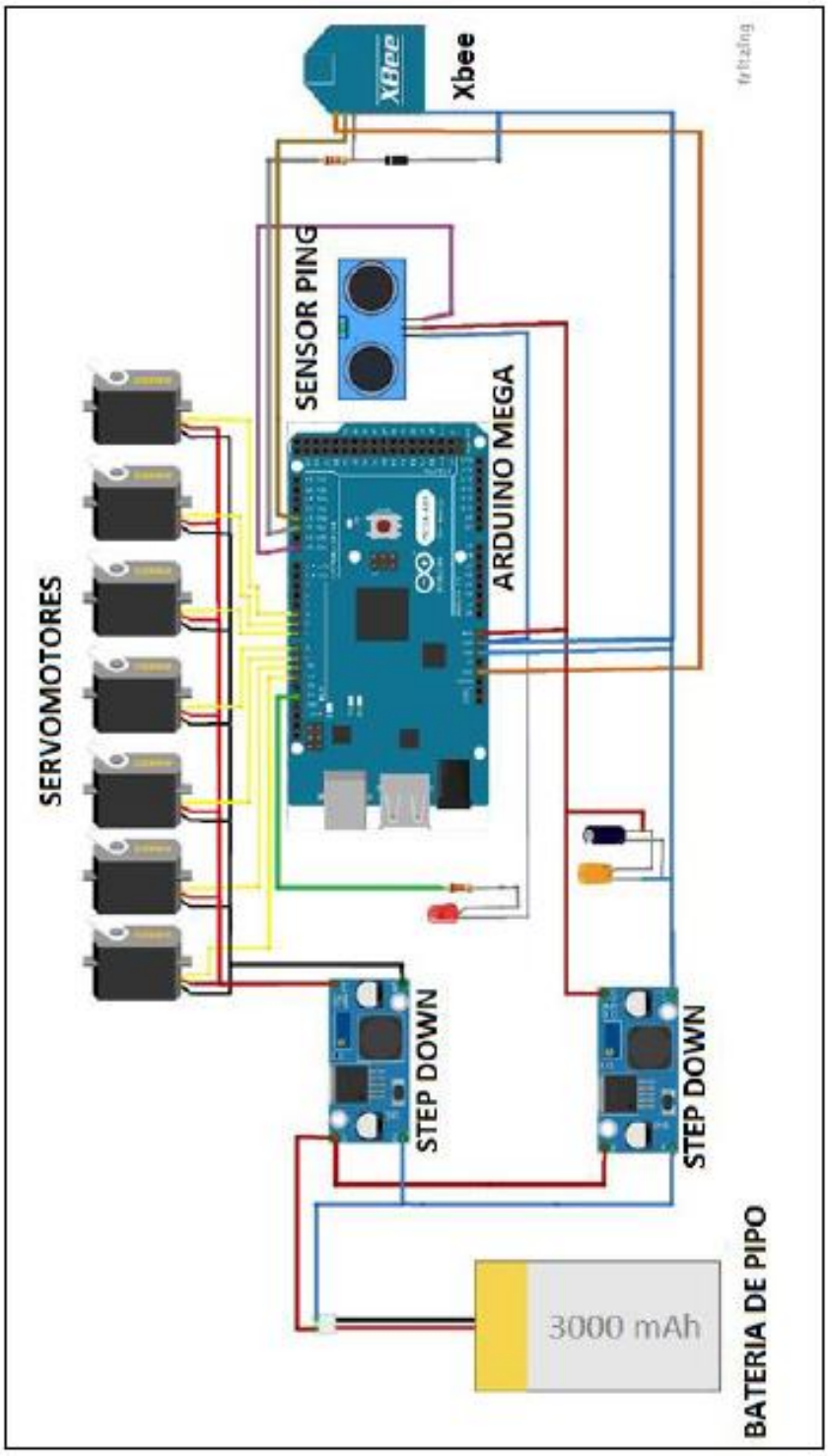


Figura 6: Diagrama esquemático
Fuente: El autor

2.1 Diagrama PCB del circuito

En la figura 7 se muestra el diagrama PCB para el circuito electrónico.

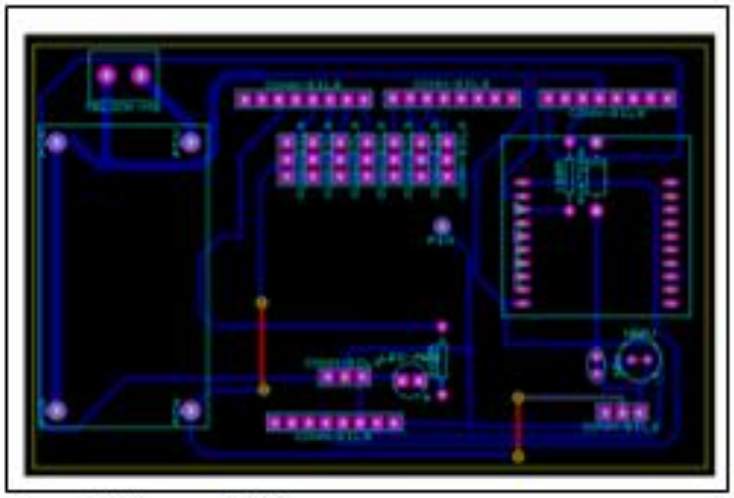


Figura 7: Diagrama PCB
Fuente: El autor

2.2 Circuito impreso

En la figura 8 se muestra el circuito impreso para el circuito electrónico.

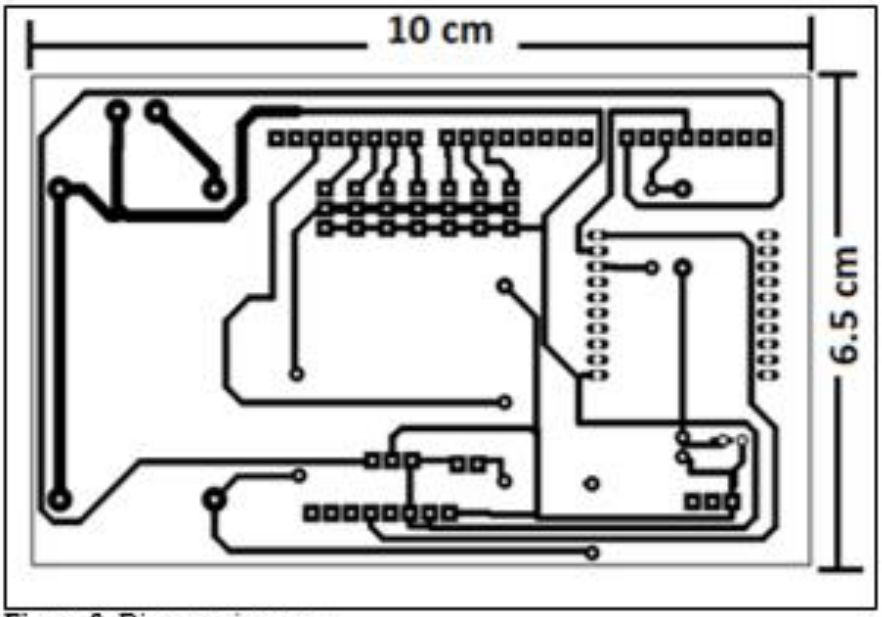


Figura 8: Diagrama impreso
Fuente: El autor

2.3 Circuito de control

En la figura 9 se muestra el circuito electrónico para el robot bípedo.

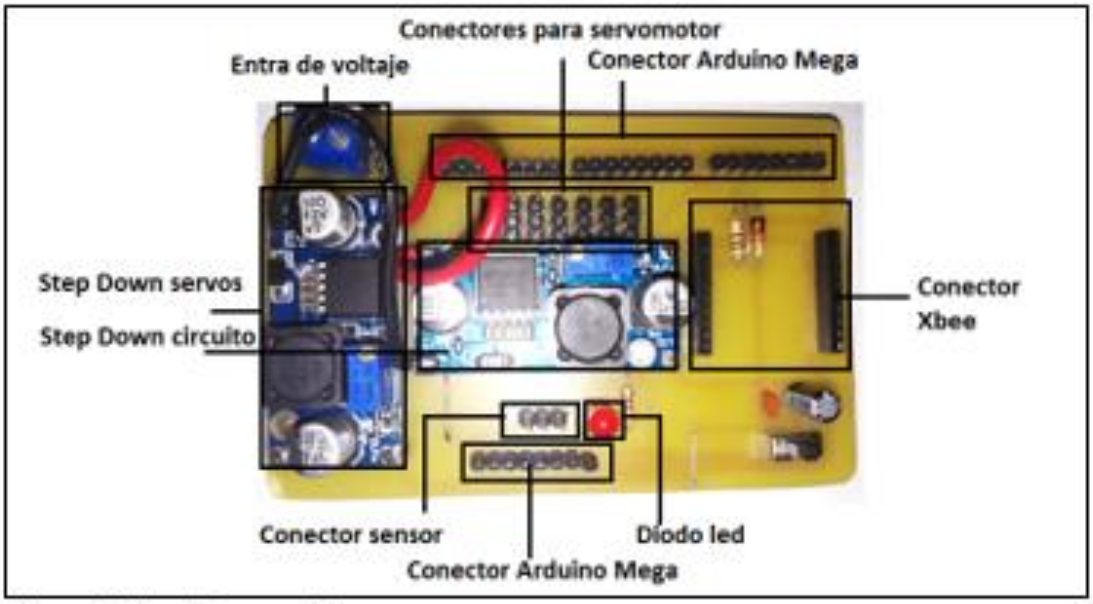


Figura 9: Circuito de control
Fuente: El autor

3. CONEXIÓN DE PUERTOS

En esta parte se describe la conexión de los puertos de la tarjeta electrónica para la batería, los servomotores y del sensor de detección.

3.1 Puerto para la batería

El circuito cuenta con una bornera de dos entradas como se observa en la figura 10. Este componente permite conectar los cables de la batería.

Figura 10: Puertos para el convertidor de voltaje
Fuente: Elaborado por el autor

3.2 Puerto para el sensor

El circuito cuenta con un conector tipo espadín de tres pines, el cual se utiliza para conectar los cables del sensor como se muestra en la figura 11.

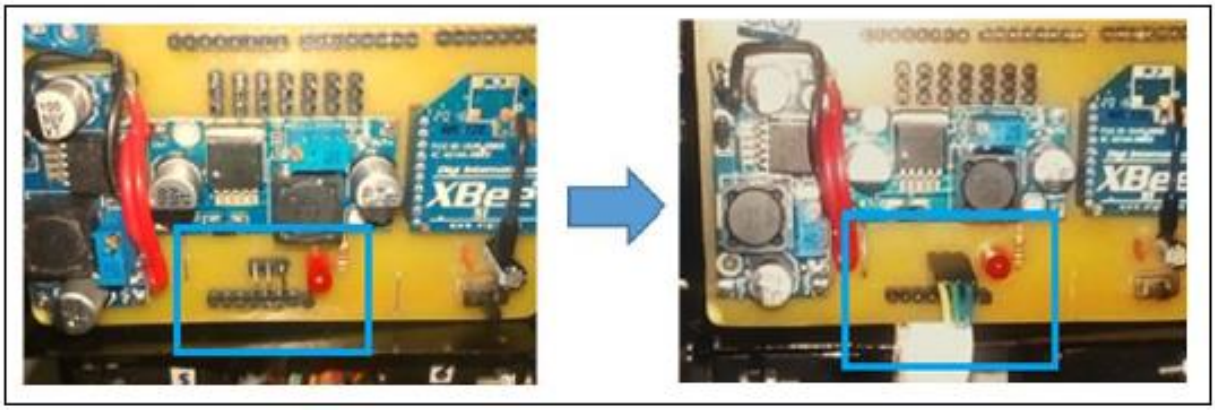


Figura 11: Puertos para el sensor ultrasónico
Fuente: Elaborado por el autor

3.3 Puestos para los servomotores

Esta conexión se realiza mediante los pines tipo espadín que se encuentran ubicados en la parte superior del circuito. Los servomotores están enumerados del uno al siete y son colocados de derecha a izquierda para reconocer la ubicación fácilmente. En la figura 12 se muestra el puerto de conexión para los servomotores.

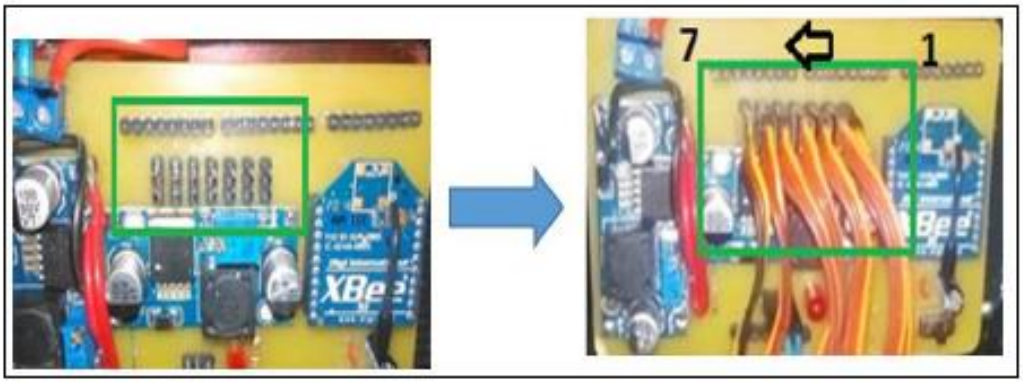


Figura 12: Montaje de la batería de lipo
Fuente: Elaborado por el autor

3.4 Pines utilizados del Arduino Mega

En la tabla 1 se observa los recursos utilizados del Arduino Mega según sus pines y su función.

Tabla 1
Pines utilizados del Arduino Mega

Pin	Componente	Función	Designación
5	Servo 1	Tobillo izquierdo	Salida
6	Servo 2	Tobillo derecho	Salida
7	Servo 3	Rodilla izquierda	Salida
8	Servo 4	Rodilla derecha	Salida
9	Servo 5	Pierna izquierda	Salida
10	Servo 6	Pierna derecha	Salida
11	Servo 7	Cabeza	Salida
14	Sensor Ping	Detección	E/S
13	Diodo led	Indicador	Salida
16	TX2	Transmisor	Entrada
17	Rx2	Receptor	Entrada
3V3	Voltaje 3 voltios	Fuente para Xbee	Entrada
GND	Tierra	Referencia 0V	Salida
VIN	Voltaje entrada	Alimentación 5V	Entrada

Fuente: Elaborado por el autor

4. ESPECIFICACIÓN DE MATERIALES

Para el desarrollo de este proyecto se utilizó los materiales que se muestran en la tabla 2.

Tabla 2
Materiales electrónicos utilizados

DESCRIPCIÓN	SÍMBOLO	CARACTERÍSTICAS TÉCNICAS
Sensor Ping Parallax		Sensor ultrasónico
LM2596		Dispositivo regulador de voltaje de 5v
Diodo led		Emisor de luz o indicador de luz
Interruptor		Dispositivo de apertura o cierre del flujo de energía
Condensador 4.7uf		Filtro para moderar el voltaje de salida
Resistencia		Se opone al paso de la corriente eléctrica
Condensador cerámico 20 pF		Filtro para altas frecuencias
Borneras 2 pines		Conector para la fuente de alimentación externa
Espadín hembra y macho		Conector para el programador Xbee
Servomotores		Motor de corriente continua.
Batería de Lipo		Fuente que almacena voltaje DC
Módulo Xbee		Dispositivo electrónico de comunicación inalámbrica
Arduino Mega		Dispositivo de control electrónico

Fuente: Elaborado por el autor

ANEXO 3

PROGRAMACIÓN EN ARDUINO

Asignación pines para servos

```
#include <Servo.h>

Servo s1; // pie izquierdo
Servo s2;  // pie derecho
Servo s3;  // rodilla izquierda
Servo s4;  //  rodilla derecha
Servo s5;  // cintura izq
Servo s6; //  cintura dere
Servo s7;  //  servo sensor
const int pingPin = 14;
```

Position central para los servomotores

```
int pos1=80;
int pos2=105;
int pos3=130;
int pos4=100;
int pos5=110;
int pos6=65;
int pos7=142;
int st=0;
```

Variables para la distancia de detección

```
 long duration, inches, cm,cmi,cmd;
 long cm1,cm2;
```

Configuración de la velocidad de comunicación

```
char rxd;
void setup() {
Serial.begin(9600);
Serial2.begin(9600);
```

Asignación de entradas y salidas

```
pinMode(A1,INPUT);//config el A1 como entrada
pinMode(A2,INPUT);
pinMode(A3,INPUT);

pinMode(A4,OUTPUT);// config el A4 salida para enviar dato del sensor

//digitalWrite(A5,OUTPUT);
pinMode(13,OUTPUT); // led rojo
```

Enlace de pines a los servomotores

```
s1.attach(5); //el servo1 esta conectado al pin D5 del arduino uno
s2.attach(6);
s3.attach(7);
s4.attach(8);
s5.attach(9);
s6.attach(10);
s7.attach(11);
```

Subrutina centrado de servomotores

```
s1.write(pos1);// pie izquierdo        0 180

s2.write(pos2);// pie derecho
s3.write(pos3);// rodilla izquierda
s4.write(pos4);// rodilla derecha
s5.write(pos5);//pierna izquierda
s6.write(pos6);//pierna derecha
s7.write(pos7);//cabeza
delay(2000);
//Serial2.write("INICIO");
}
```

Datos para definición de subrutinas

```
void loop() {
  //Serial2.write(rxd);
  digitalWrite(13,LOW);
if(rxd=='A'){adelante();}
if(rxd=='B'){parado();}
if(rxd=='C'){izquierda();}
if(rxd=='D'){derecha();}
if(rxd=='E'){navegar();}
//if(rxd=='F'){accion1();}
//if(rxd=='G'){accion2();}
while(Serial2.available()){
rxd=Serial2.read();
Serial.write(rxd);

 }
```

Subrutina adelante

```
void adelante(){
  for(int x=0;x<1;x++){

s1.write(78);//pie izquierdo
delay(20);
      s2.write(104);//tobillo derecha
      delay(20);
      s2.write(102);//tobillo derecha
      delay(20);
s1.write(76);//pie izquierdo
delay(20);
      s2.write(100);//tobillo derecha
      delay(20);
       s2.write(98);//tobillo derecha
      delay(20);
s1.write(74);//pie izquierdo
delay(20);
      s2.write(96);//tobillo derecha
      delay(20);
      s2.write(94);//tobillo derecha
      delay(20);
s1.write(72);//pie izquierdo
delay(20);
      s2.write(92);//tobillo derecha
      delay(20);
      s2.write(90);//tobillo derecha
      delay(20);
s1.write(70);//pie izquierdo
delay(20);

}
}
```

Subrutina izquierda

```
void izquierda(){
//parado();
//for(int x=0;x<3;x++){

s1.write(78);//pie izquierdo
delay(20);
      s2.write(104);//tobillo derecha
      delay(20);
      s2.write(102);//tobillo derecha
```

```
      delay(20);
s1.write(76);//pie izquierdo
delay(20);
      s2.write(100);//tobillo derecha
      delay(20);
       s2.write(98);//tobillo derecha
      delay(20);
s1.write(74);//pie izquierdo
delay(20);
}
```

Subrutina derecha

```
void derecha(){
 // parado();
//for(int x=0;x<3;x++){

//¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡¡
s2.write(105);//tobillo derecha
delay(20);
         s1.write(80);//pie izquierdo
         delay(20);
s2.write(106);//tobillo derecha
delay(20);
         s1.write(82);//pie izquierdo
         delay(20);
s2.write(108);//tobillo derecha
delay(20);
         s1.write(84);//pie izquierdo
         delay(20);
s2.write(110);//tobillo derecha
delay(20);
}
```

Subrutina Modo Automático

```
void navegar(){//  AUTOMATICO
sensor();
cm1=cm;
sensor();
cm2=cm;
cm=(cm1+cm2)/2;
if(cm>25.0){
```

```
adelante();
parado();
digitalWrite(13,LOW);
digitalWrite(A4,LOW);
Serial2.print('0');
}else{// si es que hay un objeto
digitalWrite(13,HIGH);
digitalWrite(A4,HIGH);
Serial2.print('1');

s7.write(90);//cabeza
delay(1000);
sensor();
cm1=cm;
sensor();
cm2=cm;
cm=(cm1+cm2)/2;

cmi=cm;
delay(500);
s7.write(165);//cabeza
delay(1000);
sensor();
cm1=cm;
sensor();
cm2=cm;
cm=(cm1+cm2)/2;

cmd=cm;
delay(500);
s7.write(140);//cabeza
if(cmi>cmd){
  izquierda();izquierda();
  parado();
  delay(1000);
}
if(cmd>cmi){
  derecha();derecha();
  parado();
  delay(1000);
}
}// FIN CASO CONTRARIO
}
```

Parámetros para el funcionamiento del sensor

```
void sensor(){

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH pulse
  // whose duration is the time (in microseconds) from the sending of the ping
  // to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

   cm = microsecondsToCentimeters(duration);

 // Serial2.print(cm);
  //Serial2.print("cm");
  //Serial2.println();
}

long microsecondsToCentimeters(long microseconds) {
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the object we
  // take half of the distance travelled.
  return microseconds / 29 / 2;
}
```

ANEXO 4

PROGRAMACIÓN EN VISUAL STUDIO

Dimensionar de variables

```
Public Class Form1
    Dim adelante, izquierda, derecha, detener, accion1, accion2, automatico, manual As
Integer
    Dim conectado As Boolean = False
```

Condición para establecer el enlace de comunicación

```
    Private Sub Button7_Click(sender As Object, e As EventArgs) Handles Button7.Click
        If conectado = False Then
            If Conectar() = False Then
                MsgBox("No se ha realizado la conexión al dispositivo, revise la
configuración.", MsgBoxStyle.Critical)
            End If
        Else
            If SerialPort1.IsOpen Then
                Cerrar_Conexion()
            End If
        End If
    End Sub


    Private Sub Cerrar_Conexion()
        SerialPort1.Close()
        ToolStripStatusLabel1.Text = "Desconectado"
        Label1.ForeColor = Color.DimGray
        Button7.Text = "Conectar"
        GroupBox1.Enabled = False
        Timer1.Enabled = False
        conectado = False
    End Sub
```

Envió del carácter A

```
    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        'adelante = Math.Abs(adelante - 1)
        SerialPort1.Write("A")

    End Sub
```

Envió del carácter C

```
    Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
        'izquierda = Math.Abs(izquierda - 1)
        SerialPort1.Write("C")
    End Sub
```

Envió del carácter D

```
    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
        'derecha = Math.Abs(derecha - 1)
        SerialPort1.Write("D")
    End Sub
```

Envió del carácter B

```
    Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
        'detener = Math.Abs(detener - 1)
        SerialPort1.Write("B")
    End Sub
```

Envió del carácter F

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        SerialPort1.Write("F")
        'accion1 = Math.Abs(accion1 - 1)
    End Sub
```

Envió del carácter G

```
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        'accion2 = Math.Abs(accion2 - 1)
        SerialPort1.Write("G")
    End Sub
```

Condición para la selección del modo manual o automático.

```
    Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles ComboBox1.SelectedIndexChanged
        Label2.Text = ComboBox1.SelectedItem
        If (Label2.Text = "CONTROL MANUAL") Then
        End If

        If (Label2.Text = "CONTROL AUTOMATICO") Then
            SerialPort1.Write("E")
        End If
    End Sub
Configuracion de botones

    Private Sub ConfiguraciónToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles ConfiguraciónToolStripMenuItem.Click

    End Sub

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ComboBox1.SelectedIndex = 0
    End Sub

Configuracion de botones
```

```
    Private Sub SalirToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
SalirToolStripMenuItem.Click
        If SerialPort1.IsOpen Then
            SerialPort1.Close()
        End If
        Application.Exit()
    End Sub

    Private Sub ConexiónToolStripMenuItem_Click(sender As Object, e As EventArgs)
Handles ConexiónToolStripMenuItem.Click
        Configuracion.ShowDialog()
    End Sub
```

Parámetros para la comunicación serial

```
    Function Conectar() As Boolean
        Conectar = False

        If Not SerialPort1.IsOpen Then
            With SerialPort1
                .PortName = My.Settings.puerto_com
                .BaudRate = My.Settings.baudrate
                .StopBits = IO.Ports.StopBits.One
                .DataBits = 8

                Try
                    .Open()
                    ToolStripStatusLabel1.Text = "Conectado"
                    Button7.Text = "Desconectar"
                    conectado = True
                    GroupBox1.Enabled = True
                    Timer1.Enabled = True
                    Conectar = True
                    'SerialPort1.Write("1")
                Catch ex As Exception
                    MsgBox(ex.Message, MsgBoxStyle.Critical)
                    Cerrar_Conexion()
                End Try
            End With
        End If

        Return Conectar
    End Function
```

Envió de datos por comunicación serial

```
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
        If SerialPort1.IsOpen Then
            ' REALIZA EL ENVIO DE DATOS
            'Dim envio As String = adelante.ToString + "*" + derecha.ToString + "*" +
izquierda.ToString +
            '    "*" + detener.ToString + "*" + accion1.ToString + "*" +
accion2.ToString +
            '  "*" + automatico.ToString + "*" + manual.ToString
            'SerialPort1.Write(envio)
            'Label3.Text = Trim(envio)
```

```
            ' REALIZA LA LECTURA DE DATOS
            Dim data As String = SerialPort1.ReadExisting()
            If (Trim(data) <> "") Then
                'Label3.Text = Trim(data)
                Sensor_Evasion(data)
            End If
        End If
    End Sub

    Private Sub Sensor_Evasion(data As String)
        If IsNumeric(data) Then
            Dim valor As Integer = CInt(data)

            If valor = 1 Then
                Label1.ForeColor = Color.DarkOrange
            ElseIf valor = 0 Then
                Label1.ForeColor = Color.DimGray
            End If
        End If
    End Sub
End Class
```

ANEXO 5

GUÍA PARA PROGRAMAR EN ARDUINO

MANUAL DE LENGUAJE DE PROGRAMACIÓN DE ARDUINO

INTRODUCCIÓN

Puede dividirse en: ESTRUCTURA, VARIABLES, FUNCIONES.

I. ESTRUCTURA GENERAL DE UN SKETCH/PROGRAMA

1.1. ESTRUCTURA BÁSICA DEL PROGRAMA:

Se compone de tres secciones principales:

1.1.1. Sección de declaración de variables: Ubicadas al inicio.

1.1.2. Sección llamada "*void setup*":

Delimitada por llaves de apertura y cierre. Es la primera función a ejecutar en el programa. Se ejecutan una única vez en el momento de encender o resetear la placa ARDUINO.
Se usa para inicializar:

- Los modos de trabajo de los pins E/S (*PinMode*)
- Comunicación en serie

Ejemplo:

```
int buttonPin = 3;
void setup()                    // Se ejecuta una zona vez, cuando el programa inicia
{
 Serial.begin(9600);            // Comunicación en serie
 pinMode(buttonPin, INPUT);     // Configura en pin digita l#3 como ENTRADA de datos
}
void loop()
{
 // ...
}
```

1.1.3. Sección llamada "void loop()":

Delimitada por llaves de apertura y cierre; incluye el código que se ejecuta continuamente leyendo entradas, activando salidas, etc. Esta función es el núcleo de todos los programas ARDUINO y hace la mayor parte del trabajo. Se ejecutan justo después de la sección "void setup()" infinitas veces hasta que la placa se apague (o se resetee).

Ejemplo:

```
Int pin = 10;
void setup()
 {
pinMode(pin,  OUTPUT);          // Establece  'pin'  como salida
}
void loop()
{
digitalWrite(pin, HIGH);        // pone en uno (on, 5v) el ´pin´
delay(1000);                    // espera un segundo (1000 ms)
digitalWrite(pin, LOW);         // pone en cero (off, 0v.) el ´pin´
delay(1000);                    // espera un segundo (1000 ms)
}
```

MANUAL DE LENGUAJE DE PROGRAMACIÓN DE ARDUINO

1.2. ESTRUCTURA DE CONTROL:

BLOQUES CONDICIONALES

1.2.1. Bloque "if" , "if/else" y "If/else if" :

- Las sentencias **if** comprueban si cierta condición ha sido alcanzada y ejecutan todas las sentencias dentro de las llaves si la declaración es cierta. Si es falsa el programa ignora la sentencia.

Ejemplo:

```
byte x=50;
void setup() {
        Serial.begin(9600);
        if (x >=10 && x <=20){
        Serial.println("Frase 1");
        }
        if (x >=10 | | x <=20) {
        Serial.println("Frase 2");
        }
        if (x >=10 && !(x<=20)) {
        Serial.println("Frase 3");
        }
}
        void loop(){
}
```

Veremos que solamente se muestran la "Frase 2" y la "Frase 3". Esto es debido a que la condición del primer "if" es falsa: en él se comprueba si la variable "a" es mayor que 10 Y A LA VEZ si es menor de 20. En cambio, la condición del segundo "if" es cierta: en él se comprueba si la variable "a" es mayor que 10 O BIEN menor de 20: como ya se cumple una de las condiciones –la primera–, la condición total ya es cierta valga lo que valga el resto de condiciones presentes. La condición del tercer "if" también es cierta: en él se comprueba si la variable "a" es mayor que 10 y a la vez, gracias al operador "!", que NO sea menor de 20.

- Las sentencias **if/else** permiten un mayor control sobre el flujo del código que la declaración *if* básica, por permitir agrupar múltiples comprobaciones. *"Si esto no se cumple haz esto otro".*

Ejemplo:

```
String cad1="hola";
String cad2="hola";
void setup() {
        Serial.begin(9600);
}
        void loop() {
        if (cad1 == cad2){
        Serial.println("La cadena es
igual");
}
else
{
        Serial.println("La cadena es
diferente");
}
}
```

Hay que tener en cuenta que solamente se pueden utilizar los operadores de comparación cuando ambas cadenas han sido declaradas como objetos String: si son arrays de caracteres no se pueden comparar. En este sentido, comparar si una cadena es mayor o menor que otra simplemente significa evaluar las cadenas en orden alfabético carácter tras carácter (por ejemplo, "alma" sería menor que "barco", pero "999" es mayor que "1000" ya que el carácter "9" va después del "1"). Recalcar que las comparaciones de cadenas son case-sensitive (es decir, el dato de tipo String "hola" no es igual que el dato de tipo String "HOLA").

MANUAL DE LENGUAJE DE PROGRAMACIÓN DE ARDUINO

- Las sentencias **if/else if** pueden realizar múltiples comprobaciones en una misma estructura de condiciones. Cada comprobación procederá a la siguiente sólo cuando su propio resultado sea *FALSE*. Cuando el resultado sea *TRUE*, su bloque de código contenido, será ejecutado, y el programa esquivará las siguientes comprobaciones hasta el final de la estructura de comprobaciones.

Ejemplo:

```
int numero;
        void setup(){
        Serial.begin(9600);
        }
void loop(){
        if (Serial.available() > 0) {   //El nº introducido ha de estar en el rango del tipo "int"
        numero=Serial.parseInt();
        if (numero == 23){
        Serial.println("Numero es igual a 23");
        }
        else if (numero < 23) {
                Serial.println("Numero es menor que 23");
                }
                else {
                Serial.println("Numero es mayor que 23");
                }
        }
}
```

En el código anterior aparece un primer "if" que comprueba si hay datos en el buffer de entrada del chip TTL-UART pendientes de leer. Si es así (es decir, si el valor devuelto por *Serial.available()* es mayor que 0), se ejecutarán todas las instrucciones en su interior. En cambio, si la condición resulta ser falsa (es decir, si *Serial.available()* devuelve 0 y por tanto no hay datos que leer), fijarse que la función "loop()" no ejecuta nada.

En el momento que la condición sea verdadera, lo que tenemos dentro del primer "if" es la función *Serial.parseInt()* que reconoce y extrae de todo lo que se haya enviado a través del canal serie (por ejemplo, usando el "Serial monitor") un número entero, asignándolo a la variable "numero". Y aquí es cuando llega otro "if" que comprueba si el valor de "numero" es igual a 23. Si no es así, se comprueba entonces si su valor es menor de 23. Y si todavía no es así, solo queda una opción: que sea mayor de 23. Al ejecutar este sketch a través del "Serial monitor", lo que veremos es que cada vez que enviemos algún dato a la placa, esta nos responderá con alguna de las tres posibilidades.

Nota: Ahora que ya sabemos las diferentes sintaxis del bloque "if", veamos qué tipo de condiciones podemos definir entre los paréntesis del "if". Lo primero que debemos saber es que para escribir correctamente en nuestro sketch estas condiciones necesitaremos utilizar alguno de los llamados operadores de comparación,

- ☑ x == y (x es igual a y)
- ☑ x != y (x no es igual a y)

- ☑ x < y (x es menor a y)
- ☑ x > y (x es mayor a y)
- ☑ x <= y (x es menor o igual a y)
- ☑ x >= y (x es mayor o igual a y)

1.2.2. Bloque "switch/case":

```
void setup(){
        byte x=50;
        Serial.begin(9600);
        switch (x) {
                case 20:
        Serial.println("Vale 20 exactamente");
        break;
                case 50:
        Serial.println("Vale 50 exactamente");
        break;
        default:
        Serial.println("No vale ninguna de los
valores anteriores");
        }
}
void loop(){

}
```

Consta en su interior de una serie de secciones "case" y, opcionalmente, de una sección "default". Nada más llegar a la primera línea del "switch", primero se comprueba el valor de la variable o expresión que haya entre sus paréntesis. Si el resultado es igual al valor especificado en la primera sección "case", se ejecutarán las instrucciones del interior de la misma y se dará por finalizado el "switch", en caso de no ser igual el resultado de la expresión a lo especificado en el primer "case" se pasará a comprobarlo con el segundo "case", y si no con el tercero, etc. Por último, si existe una sección "default" (opcional) y el resultado de la expresión no ha coincidido con ninguna de las secciones "case", entonces se ejecutarán las sentencias de la sección "default".

BLOQUES REPETITIVOS (BUCLES)

1.2.3. Bloque "while":

Este bucle repite la ejecución de las instrucciones que están dentro de sus llaves de apertura y cierre mientras la condición especificada entre sus paréntesis sea cierta ("true",1), sin importar el número de veces repita. *El número de iteraciones realizadas depende del estado de la condición definida.*

```
byte x=1;
void setup(){
         Serial.begin(9600);
}
void loop(){
        while (x <= 50)
        {
        Serial.println("Es menor
de 50");
        x=x+1;
        delay(250);
}
        Serial.println("Es mayor
```

Ejemplo: El "Serial monitor" los primeros 50 mensajes que aparecerán indicarán que la variable es menor de 50 porque se está ejecutando el interior del "while". Pero en el momento que la variable sea mayor (porque en cada iteración del bucle se le va aumentando en una unidad su valor), la condición del "while" dejará de ser cierta y saltará de allí, mostrando entonces el mensaje (ya de forma infinita) de que la variable es mayor que 50.

MANUAL DE LENGUAJE DE PROGRAMACIÓN DE ARDUINO

1.2.4. Bloque "do":

```
int pulsador = 24;
int led= 25;
void setup() {
 pinMode(pulsador, INPUT);
 pinMode(led, OUTPUT);
}
void loop() {
 do
 {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
 }
while (digitalRead(pulsador) == HIGH);
}
```

La condición es evaluada después de ejecutar las instrucciones escritas dentro de las llaves. Esto hace que las instrucciones siempre sean ejecutadas como mínimo una vez aun cuando la condición sea falsa, porque antes de llegar a comprobar esta, las instrucciones ya han sido leídas (a diferencia del bucle "while", donde si la condición ya de entrada era falsa las instrucciones no se ejecutaban nunca).

1.2.5. Bloque "For":

El bucle **for** es una estructura que se utiliza cuando queremos que una serie de acciones se repita un número determinado de veces, para ello se necesita de una variable índice, una condición y un incrementador. Por tanto, usaremos el bucle "for" para ejecutar un conjunto de instrucciones (escritas dentro de llaves de apertura y cierre) un número concreto de veces y estas son:

```
for (valor_inicial_contador;condicion_final;incremento){
//Instrucciones que se repetirán un número determinado de veces
}
```

- ✓ **Valor inicial del contador:** en esta parte se asigna el valor inicial de una variable entera que se utilizará como contador en las iteraciones del bucle. *Por ejemplo, si allí escribimos x=0 , se fijará la variable "x" a cero al inicio del bucle. A partir de entonces, a cada repetición del bucle, esta variable "x" irá aumentando (o disminuyendo) progresivamente de valor.*

- ✓ **Condición final del bucle:** en esta parte se especifica una condición. Justo antes de cada iteración se comprueba que sea cierta para pasar a ejecutar el grupo de sentencias internas. Si la condición se evalúa como falsa, se finaliza el bucle "for". *Por ejemplo, si allí escribimos x<10, el grupo interior de sentencias se ejecutará únicamente cuando la variable "x" valga menos de 10.*

- ✓ **Incremento del contador**: en la última de las tres partes es donde se indica el cambio de valor que sufrirá al inicio de cada iteración del bucle la variable usada como contador. Este cambio se expresa con una asignación (x++,x=x+1).

MANUAL DE LENGUAJE DE PROGRAMACIÓN DE ARDUINO

Ejemplos:

Un led parpadee 5 veces tras accionar un pulsador

```
int pulsador = 24;
int led= 25;
void setup()
{
pinMode(pulsador, INPUT);
pinMode(led, OUTPUT);
}
void loop()
{
 if (digitalRead(pulsador) == HIGH)
 {
  for (int i = 0; i<=4; i++)
  {
   digitalWrite(led, HIGH);
   delay(1000);
   digitalWrite(led, LOW);
   delay(1000);
  }
 }
else
 {
  digitalWrite(led, LOW);
 }
}
```

```
byte suma=0;
byte x;
void setup(){
   Serial.begin(9600);
   for (x=0;x<7;x=x+1){ //Sumo 7 veces el
número 5
  suma= suma + 5;
 }
}
  Serial.println(suma);
  Serial.println(suma/7); //Esto es la media
}
void loop(){}
```

En la primera repetición del bucle "for" se le asigna el valor del primer dato sumar; en la segunda repetición a ese valor de "suma" se le suma el segundo valor (por lo que en ese momento "suma" vale la suma de los dos primeros valores). En la tercera repetición se le añade el tercer valor a esa suma parcial (por lo que en ese momento "suma" vale la suma de los tres primeros valores), en la cuarta repetición el cuarto valor, y así y así hasta que se acaba de recorrer el bucle "for" y todos los datos se han ido añadiendo uno tras otro hasta conseguir la suma total. Finalmente, muestro el resultado, y muestro también la media, que no es más que esa suma total entre el número de datos introducidos en la suma.

```
byte x;
void setup(){
  Serial.begin(9600);
  for (x=0;x<10;x=x+1){
  delay(500);
 Serial.println(x);
 }
}
void loop(){}
```

La sentencia "for" primero se inicializa el contador (x=0) y seguidamente se comprueba la condición. Como efectivamente, x<10, se ejecutará la –única en este caso– sentencia interior, que muestra el propio valor del contador por el "Serial monitor". Justo después, se realiza el incremento (x=x+1), volviéndose seguidamente a comprobar la condición. Si "x" (que ahora vale 1) sigue cumpliendo la condición (sí, porque 1<10), se volverá a ejecutar la instrucción interna, mostrándose ahora el valor "1" en el "Serial monitor". Justo después de esto se volverá a realizar el incremento (valiendo x ahora por tanto 2) y se volverá a comprobar la condición, y así y así hasta que llegue un momento en el que la condición resulte falsa (cuando "x" haya incrementado tanto su valor que ya sea igual o mayor que 10), momento en el cual se finalizará el "for" inmediatamente.

NOTA: Se presenta una tabla de "equivalencia" de algunos de los operadores compuestos.

x++	Equivale a x=x+1 (Al operador "++" se le llama operador "incremento")
x--	Equivale a x=x-1 (Al operador "--" se le llama operador "decremento")
x+=3	Equivale a x=x+3
x-=3	Equivale a x=x-3

```
int led;
void setup(){
 for(led=24;led<31;led++){
   pinMode(led,OUTPUT);
 }
}
 void loop(){
  for(led=24;led<31;led++){
    digitalWrite(led,HIGH);
    delay(1000);
    //digitalWrite(led,LOW);
  }
  for(led=30;led>23;led--){
    //digitalWrite(led,HIGH);
    //delay(1000);
    digitalWrite(led,LOW);
    delay(1000);
  }
 }
```

INSTRUCCIONES

1.2.6. Instrucción "break": Ejemplo

```
int x;
void setup() {
  Serial.begin(9600);
}
void loop(){
  while (Serial.available() > 0)
 {
 char dato = Serial. read();
      if (dato =='X') break;  //termina rutina al leer una letra X
 switch (dato)
     {
     case 'R':
          Serial.println("Se eligio el color rojo");
     break;
     case 'G':
         Serial.println("Se eligio el color verde");
     break;
     case 'B':
        Serial.println("Se eligio el color azul");
    break;
    }
  }
}
```

Es usada para salir de los siguientes bloques de bucle: "do","for" y "while", "switch", sin tener que esperar a que termine el bloque de instrucciones o a que deje de cumplirse la condición lógica.

Observar que ninguna de ellas incorpora paréntesis, pero como cualquier otra instrucción, en nuestros sketches deben ser finalizadas con un punto y coma.

Debe estar escritas dentro de las llaves que delimitan las sentencias internas de un bucle.

MANUAL DE LENGUAJE DE PROGRAMACIÓN DE ARDUINO

1.2.7. Instrucción "continue":

También debe estar escrita dentro de las llaves que delimitan las sentencias internas de un bucle y sirve para finalizar la iteración actual y comenzar inmediatamente con la siguiente. Es decir, esta instrucción forzará al programa a "volver para arriba" y comenzar la evaluación de la siguiente iteración aun cuando todavía queden instrucciones pendientes de ejecutar en la iteración actual. En caso de haber varios bucles anidados (unos dentro de otros) la sentencia "continue" tendrá efecto únicamente en el bucle más interior de ellos.

Ejemplo:

```
byte x;
void setup(){
   Serial.begin(9600);
   for(x=0;x<10;x++){
      if (x==4){
      break;        //continue;
      }
  Serial.println(x);
  }
}
void loop(){}
```

¿Qué pasaría si sustituimos la instrucción ***break*** por la sentencia ***continue***, dejando el resto del código anterior exactamente igual? Que veríamos una lista de números desde el 0 hasta el 9, excepto precisamente el 4. Esto es así porque la instrucción continue interrumpe la ejecución de la iteración en la cual "x" es igual a 4 (y por tanto, la instrucción Serial.println() correspondiente no se llega a ejecutar) pero continúa con la siguiente iteración del bucle de forma normal, en la cual a "x" se le asigna el valor 5.

1.2.8. Instrucción "goto": Ejemplo

```
void setup(){
Serial.begin(9600);
}
void loop()
{
 inicio:
Serial.println("Ciclo infinito");
delay(500);
goto inicio;
}
```

Ésta es la instrucción que te otorga más libertad, pero con la que más cuidado hay que tener, se escribe una etiqueta en cualquier parte del programa, y al usar la instrucción: goto etiqueta el programa inmediatamente saldrá de cualquier estructura de selección o iteración para dirigirse a la etiqueta seleccionada.

1.2.9. Instrucción "return":

Ésta instrucción nos devolverá algo de una función (que no sea void). Lo que escribamos por debajo de return en la función donde lo usemos, no se ejecutará nunca. Ya que, cuando llega a return, vuelve a la función que lo llamó.

Hemos visto las funciones " void " (quiere decir que no devuelven ningún valor) , pero también existen las que sí lo hacen, como las " int " , " float " ,etc (vamos, que ponemos el tipo de variable del valor que queremos retornar)

Ejemplo:

```
int suma(int numero_A, int numero_B);  // declaramos nuestra función
  void setup()
 {
  Serial.begin(9600);
  int resultado = 0;
  int primerNumero = 19;
  int segundoNumero = 2;
  resultado = suma( primerNumero, segundoNumero );  // aquí vemos que el nombre de los parámetros cambió, eso es porque realmente no opera con el nombre, si no, con las posiciones
  Serial.print("El resultado es: ");
  Serial.println(resultado); // imprimimos el resultado
 }
  void loop()
 {
 }
 int suma( int numero_A, int  numero_B)  // aqui tienen que tener los mismos nombres que arriba
 {
   return numero_A + numero_B; // aquí nos devolverá la suma de los dos operandos
 }
```

1.3. ELEMENTOS DE SINTAXIS:

Para evitar errores de compilación, tener en cuenta lo siguiente:

- ✓ **" ; "(**punto y coma).- El punto y coma es uno de los símbolos más usados en C, C++; y se usa con el fin de indicar el final de una línea de instrucción.
  Ejemplo: int a = 13;

- ✓ "**{ }**" (llaves).- Las llaves sirven para definir el principio y el final de un bloque de instrucciones. Se utilizan para los bloques de programación setup(), loop(), if.., etc.
  Ejemplos:

- ✓ **/*....*/** Bloque de comentario: Son áreas de texto que pueden abarcar más de una línea, lo que escribamos entre esos símbolos será ignorado por el programa.

- ✓ **//** Línea de Comentario: Funciona como el bloque de comentario, con la diferencia que solo será ignorado el texto que este a su derecha, al cambiar de línea perderá el efecto.

- ✓ **#define**.- permite al programador dar un nombre a un valor constante antes de compilar el programa. Constantes definidas en ARDUINO no ocupan ningún espacio en el chip. El compilador de reemplazar las referencias a estas constantes con el valor definido en tiempo de compilación.

  Ejemplo:

```
#define ledPin 3 // El compilador reemplazará cualquier mención de ledPin con el valor 3 cuando se ejecute el programa.
```

MANUAL DE LENGUAJE DE PROGRAMACIÓN DE ARDUINO

- ✓ **#include**.- se utiliza para incluir bibliotecas fuera de programa. Esto le da acceso al programador para un gran grupo de bibliotecas estándar de C (grupos de funciones pre-hechos), y también bibliotecas escrito especialmente para ARDUINO.
  Ejemplo:

```
#include <avr/pgmspace.h>

prog_uint16_t myConstants[] PROGMEM = {0, 21140, 702 , 9128,  0, 25764, 8456,
0,0,0,0,0,0,0,0,29810,8968,29762,29762,4500};
```

NOTA: Tenga en cuenta que "#include", al igual que "#define", no termina en ";" (punto y coma), y el compilador producirá mensajes de error si se le agrega.

Fuente: (PUCP, 2016)

ANEXO 6

GUÍA PARA PROGRAMAR EN VISUAL STUDIO

aprenderaprogramar.com

Declaración de variables (integer, single, double, boolean, etc.) en Visual Basic. Dim. Ejemplos. (CU00309A)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº8 del Curso Visual Basic Nivel I

29

Declaración de variables en Visual Basic. Dim. Ejemplos.

DECLARACIÓN DE VARIABLES EN VISUAL BASIC

A la hora de declarar variables conviene seguir las pautas indicadas en el curso "Bases de la programación nivel I" de aprenderaprogramar.com y, resumidamente:

- Elegir siempre el tipo de variable más sencillo posible. Consideraremos que el grado de sencillez viene definido por la memoria requerida (a menor memoria requerida mayor es la sencillez). Esto redunda en un menor consumo de recursos del sistema y en una mayor rapidez de las operaciones que realiza el ordenador.
- Realizar declaraciones ordenadas y claras.
- Evitar el uso de los tipos Variant u Object en la medida de lo posible.

Visual Basic permite declarar variables en múltiples líneas una por línea pero también varias variables en una línea. Veamos ejemplos:

Expresión abreviada	Equivalente
a) Dim i%, j%, k%	a´) Dim i As Integer, j As Integer, k As Integer
b) Dim i% : Dim j% : Dim k%	b´) Dim i As Integer : Dim j As Integer : Dim k As Integer
c) Dim i%, j%, k As Integer	c´) Dim i As Integer, j As Integer, k As Integer
d) Dim i% Dim j% Dim k%	d´) Dim i As Integer Dim j As Integer Dim k As Integer
e) Dim i As Integer Dim j%, k%	e´) Dim i As Integer Dim j As Integer, k As Integer

Las opciones a), b), c), d), e) dan lugar a un mismo resultado. Se declaran tres variables tipo Integer denominadas i, j, k. Las opciones a´- e´), escritas de forma extendida, también son válidas y con el mismo resultado. Nosotros nos decantamos por usar opciones como la c), donde en una misma línea aparezcan variables de un tipo de forma abreviada, excepto la última que se pone extendida de forma aclaratoria, y en las que las variables de una misma línea tienen algún tipo de relación. Por ejemplo tratarse de contadores para bucles.

Ten en cuenta que la declaración Dim i, j, k As Integer no declara tres variables de tipo entero de forma correcta. Esta sintaxis es imprecisa y por tanto no debemos utilizarla. Según la versión de Visual Basic que estemos usando esto puede generar errores o problemas. Conviene tener cuidado a la hora de realizar declaraciones de variables para evitar prácticas de este tipo.

Una expresión como Dim i!, j%, k& equivale a Dim i As Single, j As Integer, k As Long. Esta expresión es válida, aunque como hemos dicho preferimos no declarar distintos tipos de variables en una misma línea.

Declaración de variables en Visual Basic. Dim. Ejemplos.

EJERCICIO

Realizar una declaración de variables para las siguientes propuestas de programas.

a) Un programa que muestra un mensaje de bienvenida.

b) Un programa que nos pide nuestra edad y nos muestra el año en que nacimos.

c) Un programa que nos muestra el valor medio de tres magnitudes.

d) Un programa que trabaja con: tres contadores para bucles, tres valores de tensión en una viga (de magnitud no conocida), valores de longitud de la viga, canto y ancho y un valor interruptor que permite o impide que se muestren resultados.

SOLUCIÓN

a) Dim mensaje As String. También es válida la expresión abreviada Dim mensaje$.

b) Dim edad As Integer. También podemos usar la expresión abreviada Dim edad%.

Se podría definir otra variable llamada añonacimiento según el programador lo estime oportuno o no.

c) Dim magnitud1!, magnitud2!, magnitud3 As Single

Dim media As Single

Hemos utilizado el tipo Single para cubrirnos las espaldas. No sabemos con qué tipo de valores vamos a trabajar y por ello cogemos un tipo de variables bastante amplio como es el Single. Si se considera necesario también puede usarse Double.

Hemos utilizado dos líneas pero sólo un tipo de variable ¿Por qué? El programador ordena la declaración de variables como estime más conveniente. En este caso ordenamos "valores para el cálculo" en una línea y "resultados" en otra. Resulta más fácil de leer y analizar que estando todo en una sola línea.

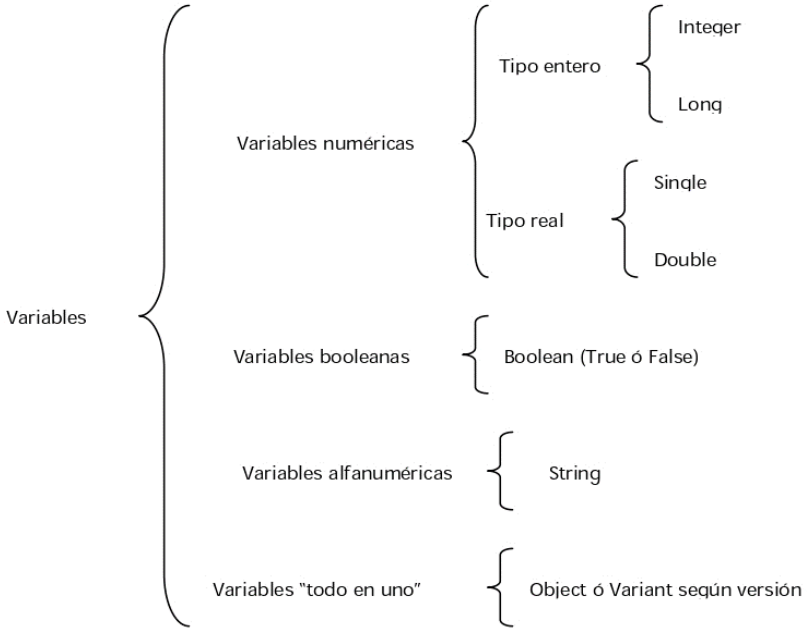
d) Dim i%, j%, k As Integer
Dim tension1!, tension2!, tension3 As Single
Dim largo!, canto!, ancho As Single
Dim Muestraresultados As Boolean.

Siempre será preferible usar las expresiones tension1, tension2 y tension3 antes que t1, t2 y t3 que resultan más confusas. Sin embargo, este tipo de decisiones siempre quedan a criterio del programador.

Declaración de variables en Visual Basic. Dim. Ejemplos.

CONTENIDO Y ASIGNACIÓN DE CONTENIDO A VARIABLES

Las normas para asignar contenido a variables serán las indicadas en el curso "Bases de la programación nivel I" de aprenderaprogramar.com, teniendo en cuenta que de forma general usaremos estas equivalencias.



Ejemplos de asignación de contenidos son estos:

Declaración	Ejemplo asignación contenidos
Dim A As Integer	A = 123
Dim A As Single	A = - 3323.57
Dim A As String	A = "Hay que mejorar"
Dim A As String	A = "123 coches"
Dim A As String	A = "Son 35,37 euros"
Dim Salario As Integer	A = 30500
Dim Salario As Single	A = 30500

Declaración de variables en Visual Basic. Dim. Ejemplos.

Declaración	Ejemplo asignación contenidos
Dim Salario As String	A = "Se asignarán 30500 euros"
Dim A%, B%, Suma%	A = 5 + 2 B = 32 Suma = A + B [Suma valdrá 39]
Dim A%, B%, C%, D%, E%	A = 5 B = 32 C = A * B [C toma el valor 160] D = A + C [D toma el valor 165] E = D [E toma el valor 165]
Dim Agotamiento As Boolean	Agotamiento = True

Nota: la separación decimal normalmente es un punto, pero tendrás que comprobarlo en tu computador porque puede variar en función de las configuraciones aplicadas. Es decir, normalmente se escribirá un decimal como 3.33 en lugar de cómo 3,33.

Un programa que conste de:

```
Dim A As Integer
A = 7 * B
```

dará lugar a un error debido a que *B* no está declarada.

En cambio:

```
Dim A%, B%
A = 7 * B
```

Supone que A valga cero, ya que si no se asigna un valor a B ésta tiene por contenido el valor por defecto, es decir, cero. 7 * B equivale en este caso a 7 * 0, que vale cero.

Próxima entrega: CU00310A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61

aprenderaprogramar.com

Ejemplo de programa básico en Visual Basic. Option Explicit, Form, Dim, String, etc. (CU00310A)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº9 del Curso Visual Basic Nivel I

29

Asignacion de contenidos en variables Visual Basic

CREAR UN PROGRAMA BÁSICO

Vamos a crear un programa que declare una variable tipo String llamada mensaje y nos muestre un texto en pantalla.

Para ello como primer paso abrimos un proyecto de Visual Basic. Recordar que esto se hace de la siguiente manera. En el menú Archivo seleccionamos la opción "Nuevo proyecto". A continuación elegimos:

a) Para las versiones que nos dan la opción "Aplicación de Windows Forms" elegimos esta opción.

b) Para las versiones que nos dan la opción "Exe estándar" elegimos esta opción.

Una vez abierto el proyecto debe aparecernos el formulario vacío sobre la pantalla. Hacemos doble click sobre el formulario y se nos debe abrir la ventana de código conteniendo algunas líneas. En caso de que no se te abra la ventana de código, vete al menú "Ver" y elige la opción "Código".

Escribiremos el siguiente código que corresponde al programa que vamos a ejecutar:

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim mensaje As String

Private Sub Form_Load()
mensaje = "Bienvenido a este programa"
MsgBox (mensaje)
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim mensaje As String
    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        mensaje = "Bienvenido a este programa"
        MsgBox(mensaje)
    End Sub
End Class
```

Ten en cuenta que según la versión de Visual Basic que estés utilizando pueden existir pequeñas diferencias en el código que debe usarse. Nosotros pretendemos centrarnos en la lógica de la programación más que en los detalles de la sintaxis de una versión concreta. Si es necesario, consulta la ayuda para conocer la sintaxis exacta que debes emplear.

Analicemos las líneas que lo componen.

1. Option Explicit (Option Explicit On). Ya hemos definido su significado.
2. Dim mensaje as String, declara una variable denominada mensaje como tipo alfanumérico de longitud variable.
3. Private Sub Form Load() ó Private Sub Form1_Load (…). Esta parte de código corresponde a la apertura de un módulo de código de Visual Basic, que en la terminología del lenguaje se denomina

Procedimiento. La cuestión se puede hacer un poco complicada en lo que se refiere a terminología, pero no vamos a preocuparnos por esta cuestión ahora.

En el curso "Bases de la programación I" de aprenderaprogramar.com se trabaja con programas que constan de un algoritmo principal y distintos módulos o subprogramas. La estructura y terminología de un programa en Visual Basic es algo más compleja, al trabajarse con distintos elementos dentro de un programa. Estos elementos varían según la versión, pero podrían ser por ejemplo módulos de formulario (.frm), módulos estándar (.bas), módulos de clases (.cls), etc.. Nosotros vamos a centrarnos en lo que es un módulo de formulario: un fichero en el que almacenamos información relacionada con el formulario donde hemos colocado objetos y para el que hemos escrito un código. Así pues, de momento para nosotros el programa va a constar de:

- Un formulario donde hemos colocado objetos (trabajo de diseño de tipo gráfico). Excepcionalmente un formulario puede carecer de objetos y encontrarse vacío.
- Un código que consta normalmente de: Option Explicit (Option Explicit On), Declaración de variables y Procedimientos. Un procedimiento es un conjunto de código que se ejecutará en un momento dado, algo equivalente a lo que habíamos definido con el pseudocódigo como módulo o subprograma.

Un procedimiento puede ejecutarse por varios motivos como:

a) Ser llamado desde alguna parte del programa.

b) Se produce un evento que da lugar a que se ejecute el código.

En este ejemplo que estamos viendo, el procedimiento se ejecuta cuando tiene lugar un evento que es la carga del formulario (form load), que vamos a considerar la ejecución del programa. Así pues, el código comprendido entre el fragmento de código de tipo "Private Sub Form_Load()" y "End Sub" se va a ejecutar cada vez que mandemos correr el programa, es decir, cada vez que ordenemos su ejecución.

4. mensaje = "Bienvenido a este programa", asigna a la variable mensaje el contenido de tipo alfanumérico indicado.
5. MsgBox (mensaje), indica al ordenador que proceda a mostrar el contenido de la variable mensaje dentro de una caja de mensajes.
6. End Sub, define el final del módulo de código (procedimiento).

Ejecuta el programa pulsando la tecla F5. También puedes hacerlo a través del menú <<Ejecutar / Iniciar>> ó <<Depurar / Iniciar la depuración>> según la versión que estés utilizando. En algunas versiones existe la opción <<Ejecutar / Iniciar con compilación completa>>, que permite que la ejecución del programa se haga realizando una revisión de errores más exhaustiva que si se ejecuta el programa pulsando simplemente F5.

Asignacion de contenidos en variables Visual Basic

Tras ejecutar el programa te aparecerá en pantalla el mensaje Bienvenido a este programa. Para cerrar el programa pulsa aceptar y haz click sobre el aspa de cierre de la ventana. Vamos a hacer una pequeña mejora estética. Vete a Ver-Diseñador (Ver-Objeto) y cambia el valor de la propiedad Caption ó Text (según la versión que estés usando) del form a "Bienvenida" y modifica el código de la siguiente manera.

Para las versiones menos recientes:

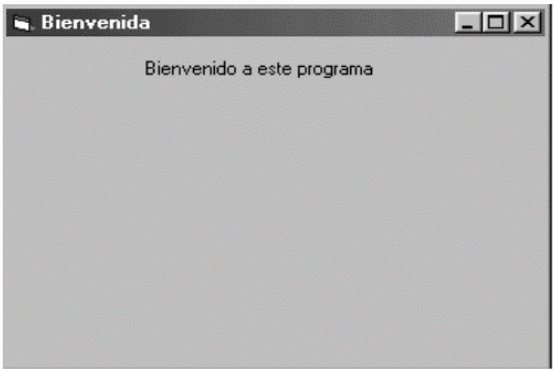
```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim mensaje As String
Dim espacios As String
Private Sub Form_Load()
    espacios = vbTab
    mensaje = espacios & "Bienvenido a
este programa" & espacios
    MsgBox (mensaje)
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
  Dim mensaje As String
  Dim espacios As String

  Private Sub Form1_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    espacios = vbTab
    mensaje = espacios & "Bienvenido a este
programa" & espacios
    MsgBox(mensaje)
  End Sub
End Class
```

Fíjate que cuando aparezca la ventana del formulario, debido a que hemos modificado sus propiedades, ahora nos aparecerá el texto “Bienvenida” en la parte superior.



También hemos usado otros elementos que iremos explicando poco a poco a lo largo del curso. El objetivo ahora es simplemente irnos familiarizando con Visual Basic.

Asignacion de contenidos en variables Visual Basic

La constante vbTab nos ha servido para introducir un tabulador. Ello da lugar a que el texto aparezca con un pequeño margen por el lado izquierdo y derecho sobre la ventanita del mensaje. Prueba a introducir variables de tipo numérico y a dar lugar a que se muestren sus contenidos sobre la ventana del mensaje. Juega con los nombres de las variables, sus contenidos y la asignación de contenidos.

También puede resultar de interés que compruebes qué ocurre si declaras una variable de un tipo y le asignas contenido de otro tipo.Por ejemplo declara *Dim salario As Integer*, asígnale el contenido *salario = "Son 3000 euros"* y prueba a mostrarlo en pantalla.

¿Qué ocurre cuando incumplimos las previsiones de *Visual Basic* para asignar contenidos a variables (por ejemplo, para una variable A tipo Integer definir A = 532000000000000000000000, que está fuera del rango previsto)? No vamos a analizar los distintos casos que se pueden presentar, sino a tratar de dar una respuesta genérica. Cuando hacemos algo no esperado, como asignar un valor fuera de rango, asignar un valor que no concuerda con el tipo de la variable, sumar variables numéricas con alfanuméricas, asignar decimales a un número entero..., etc. pueden suceder varias cosas:

a) Salta un error. Un mensaje impide que el programa comience a ejecutarse, o bien el programa se detiene mientras se está ejecutando.
b) El programa se ejecuta pero los resultados no son los deseados.
c) El programa se ejecuta y los resultados son los deseados.

En resumen, es difícil prever lo que va a suceder, aunque se pueden estudiar y manejar las circunstancias. Por ejemplo una variable que se declare como tipo Long pero a la que se asigna un contenido numérico real con dos decimales no da lugar a un error, pero sí a una pérdida de información derivada de que el número decimal se va a redondear a un entero. Las consecuencias de esta circunstancia habría que valorarlas para cada programa y circunstancias concretas.

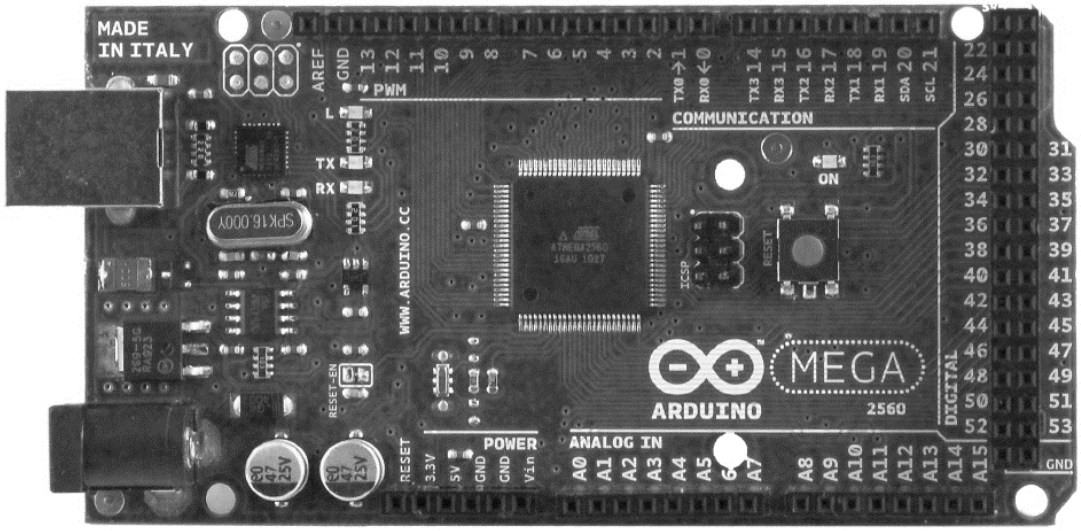
Como programadores hemos de buscar programas 100% predecibles, en los que no se pueda producir que "sea difícil prever lo que va a suceder". Por tanto intentaremos que la declaración y asignación de contenidos a variables se ajuste a las normas de *Visual Basic*. En última instancia, podremos prever una rutina de gestión de errores para casos imprevistos.

Fuente: (Rancel, 2019)

ANEXO 7

DATASHEET ARDUINO MEGA

Arduino MEGA 2560



CE

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

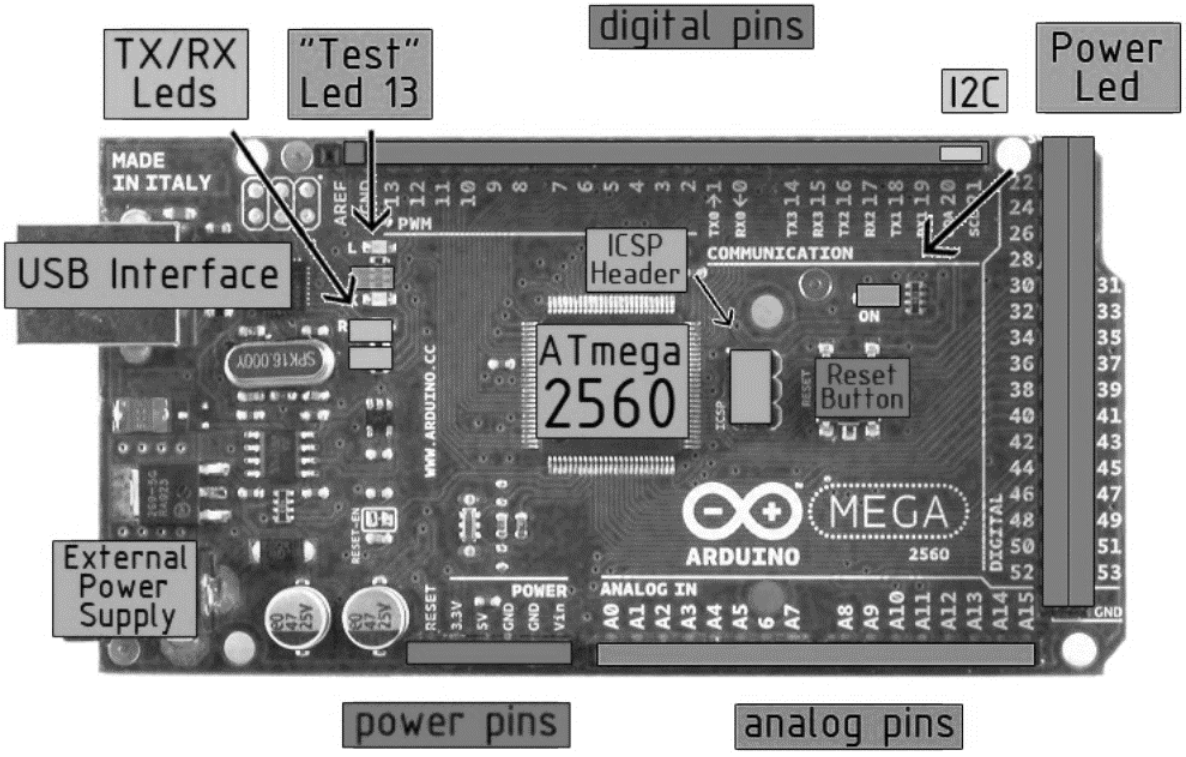
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

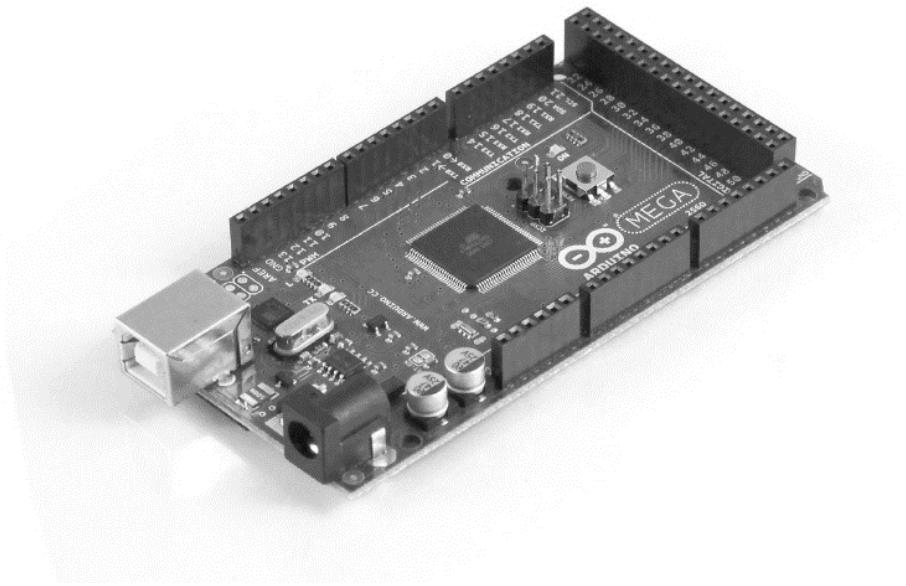
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

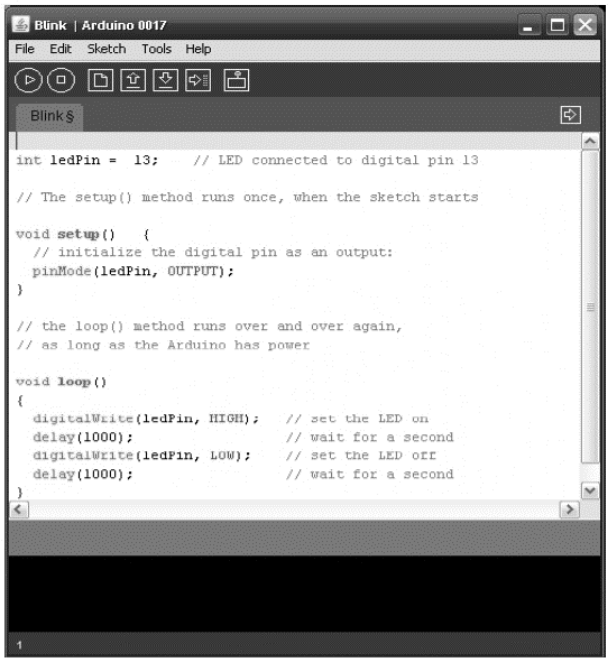
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.

In **Tools>Board** select MEGA

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.



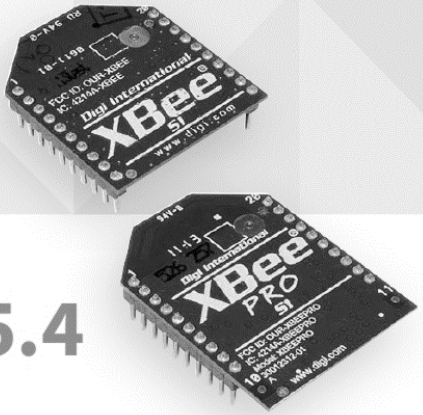
Fuente: (Robot Shop, 2019)

ANEXO 8

DATASHEET XBEE

DIGI

EMBEDDED RF
MODULES FOR OEMS



DIGI XBEE® S1 802.15.4 RF MODULES

Easy-to-deploy modules provide critical end-point connectivity to devices and sensors

Digi XBee RF modules provide OEMs with a common footprint shared by multiple platforms, including multipoint and ZigBee/Mesh topologies, and both 2.4 GHz and 900 MHz solutions. OEMs deploying the Digi XBee can substitute one Digi XBee for another, depending upon dynamic application needs, with minimal development, reduced risk and shorter time-to-market.

Digi XBee 802.15.4 RF modules are ideal for applications requiring low latency and predictable communication timing. Providing quick, robust communication in point-to-point, peer-to-peer, and multipoint/star configurations, Digi XBee 802.15.4 products enable robust end-point connectivity with ease. Whether deployed as a pure cable replacement for simple serial communication, or as part of a more complex hub-and-spoke network of sensors, Digi XBee 802.15.4 RF modules maximize performance and ease of development.

Digi XBee 802.15.4 modules seamlessly interface with compatible gateways, device adapters and range extenders, providing developers with true beyond-the-horizon connectivity.

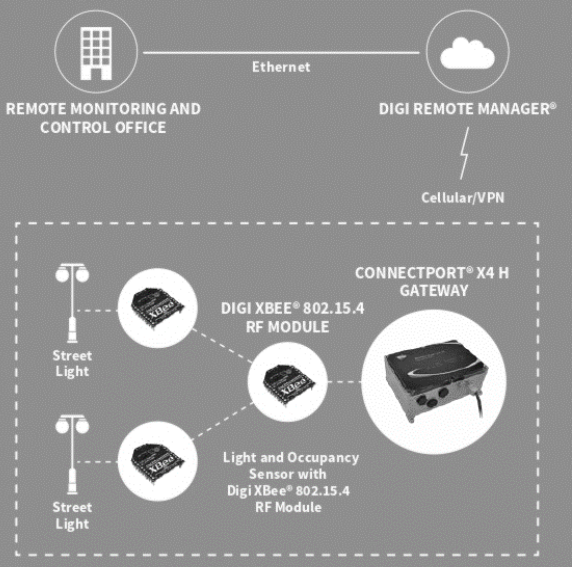
BENEFITS

- Simple, out-of-the-box RF communications, no configuration needed
- Point-to-multipoint network topology
- 2.4 GHz for worldwide deployment
- Common Digi XBee footprint for a variety of RF modules
- Low-power sleep modes
- Multiple antenna options

RELATED PRODUCTS

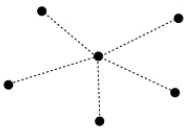
ConnectPort® X4/X4H Gateways | Digi XBee® Adapters | XCTU | Digi Remote Manager® | Development Kits

APPLICATION EXAMPLE

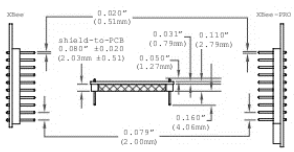


SPECIFICATIONS	Legacy Digi XBee® S1 802.15.4	Legacy Digi XBee-PRO® S1 802.15.4
PERFORMANCE		
RF DATA RATE	250 kbps	250 kbps
INDOR/URBAN RANGE	100 ft (30 m)	300 ft (100 m)
OUTDOOR/RF LINE-OF-SIGHT RANGE	300 ft (100 m)	1 mi (1.6 km)
TRANSMIT POWER	1 mW (+0 dBm)	60 mW (+18 dBm)*
RECEIVER SENSITIVITY (1% PER)	-92 dBm	-100 dBm
DIGI HARDWARE	S1	
TRANSCEIVER CHIPSET	Freescale MC13212	
FEATURES		
SERIAL DATA INTERFACE	3.3V CMOS UART	
CONFIGURATION METHOD	API or AT Commands, local or over-the-air	
FREQUENCY BAND	2.4 GHz	
INTERFERENCE IMMUNITY	DSSS (Direct Sequence Spread Spectrum)	
SERIAL DATA RATE	1200 bps - 250 kbps	
ADC INPUTS	(6) 10-bit ADC inputs	
DIGITAL I/O	8	
ANTENNA OPTIONS	Chip, Wire Whip, U.FL, & RPSMA	
NETWORKING & SECURITY		
ENCRYPTION	128-bit AES	
RELIABLE PACKET DELIVERY	Retries/Acknowledgments	
IDS AND CHANNELS	PAN ID, 64-bit IEEE MAC, 16 Channels	
POWER REQUIREMENTS		
SUPPLY VOLTAGE	2.8 - 3.4VDC	2.8 - 3.4VDC
TRANSMIT CURRENT	45 mA @ 3.3VDC	215 mA @ 3.3VDC
RECEIVE CURRENT	50 mA @ 3.3VDC	55 mA @ 3.3VDC
POWER-DOWN CURRENT	<10 uA @ 25° C	
REGULATORY APPROVALS		
FCC (USA)	OUR-Digi XBee	OUR-Digi XBeePRO
IC (CANADA)	4214A-Digi XBee	4214A-Digi XBeePRO
ETSI (EUROPE)	Yes	Yes - Max TX 10 mW
C-TICK AUSTRALIA	Yes	
TELEC (JAPAN)	Yes	

802.15.4 – Star

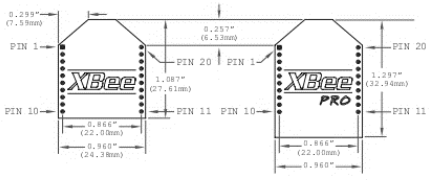


(Side Views)



(Top View)

(Top View)



PART NUMBERS	DESCRIPTION
MODULES	
XB24-AWI-001	Digi XBee S1 802.15.4 low-power module w/ wire antenna
XB24-API-001	Digi XBee S1 802.15.4 low-power module w/ PCB antenna
XB24-AUI-001	Digi XBee S1 802.15.4 low-power module w/ U.fl connector
XB24-ASI-001	Digi XBee S1 802.15.4 low-power module w/ RPSMA connector
XBP24-AWI-001	Digi XBee-PRO S1 802.15.4 extended-range module w/ wire antenna
XBP24-AWI-001J	Digi XBee-PRO S1 802.15.4 extended-range module w/ wire antenna (International)
XBP24-AUI-001	Digi XBee-PRO S1 802.15.4 extended-range module w/ U.fl connector
XBP24-AUI-001J	Digi XBee-PRO S1 802.15.4 extended-range module w/ U.fl connector (International)
XBP24-ASI-001	Digi XBee-PRO S1 802.15.4 extended-range module w/ RPSMA connector
XBP24-ASI-001J	Digi XBee-PRO S1 802.15.4 extended-range module w/ RPSMA connector (International)
XBP24-API-001	Digi XBee-PRO S1 802.15.4 extended-range module w/ PCB antenna
XBP24-API-001J	Digi XBee-PRO S1 802.15.4 extended-range module w/ PCB antenna (International)

FOR MORE INFORMATION
PLEASE VISIT WWW.DIGI.COM

Fuente: (DIGI, 2019)

ANEXO 9

DATASHEET LM2596

LM2596

3.0 A, Step-Down Switching Regulator

The LM2596 regulator is monolithic integrated circuit ideally suited for easy and convenient design of a step−down switching regulator (buck converter). It is capable of driving a 3.0 A load with excellent line and load regulation. This device is available in adjustable output version and it is internally compensated to minimize the number of external components to simplify the power supply design.

Since LM2596 converter is a switch−mode power supply, its efficiency is significantly higher in comparison with popular three−terminal linear regulators, especially with higher input voltages.

The LM2596 operates at a switching frequency of 150 kHz thus allowing smaller sized filter components than what would be needed with lower frequency switching regulators. Available in a standard 5−lead TO−220 package with several different lead bend options, and D^2PAK surface mount package.

The other features include a guaranteed ±4% tolerance on output voltage within specified input voltages and output load conditions, and ±15% on the oscillator frequency. External shutdown is included, featuring 80 μA (typical) standby current. Self protection features include switch cycle−by−cycle current limit for the output switch, as well as thermal shutdown for complete protection under fault conditions.

Features

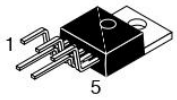
- Adjustable Output Voltage Range 1.23 V – 37 V
- Guaranteed 3.0 A Output Load Current
- Wide Input Voltage Range up to 40 V
- 150 kHz Fixed Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode, typ 80 μA
- Thermal Shutdown and Current Limit Protection
- Internal Loop Compensation
- Moisture Sensitivity Level (MSL) Equals 1
- Pb−Free Packages are Available

Applications

- Simple High−Efficiency Step−Down (Buck) Regulator
- Efficient Pre−Regulator for Linear Regulators
- On−Card Switching Regulators
- Positive to Negative Converter (Buck−Boost)
- Negative Step−Up Converters
- Power Supply for Battery Chargers

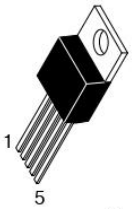
ON Semiconductor®

http://onsemi.com



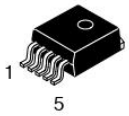
TO−220
TV SUFFIX
CASE 314B

Heatsink surface connected to Pin 3



TO−220
T SUFFIX
CASE 314D

Pin
1. V_{in}
2. Output
3. Ground
4. Feedback
5. $\overline{ON}$/OFF



D^2PAK
D2T SUFFIX
CASE 936A

Heatsink surface (shown as terminal 6 in case outline drawing) is connected to Pin 3

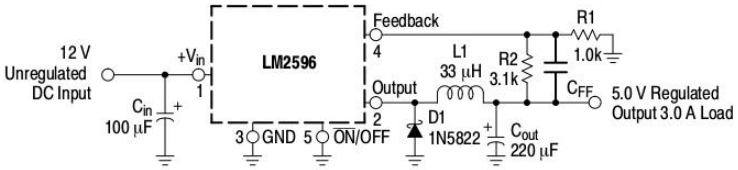
ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 23 of this data sheet.

DEVICE MARKING INFORMATION

See general marking information in the device marking section on page 23 of this data sheet.

Typical Application (Adjustable Output Voltage Version)



Block Diagram

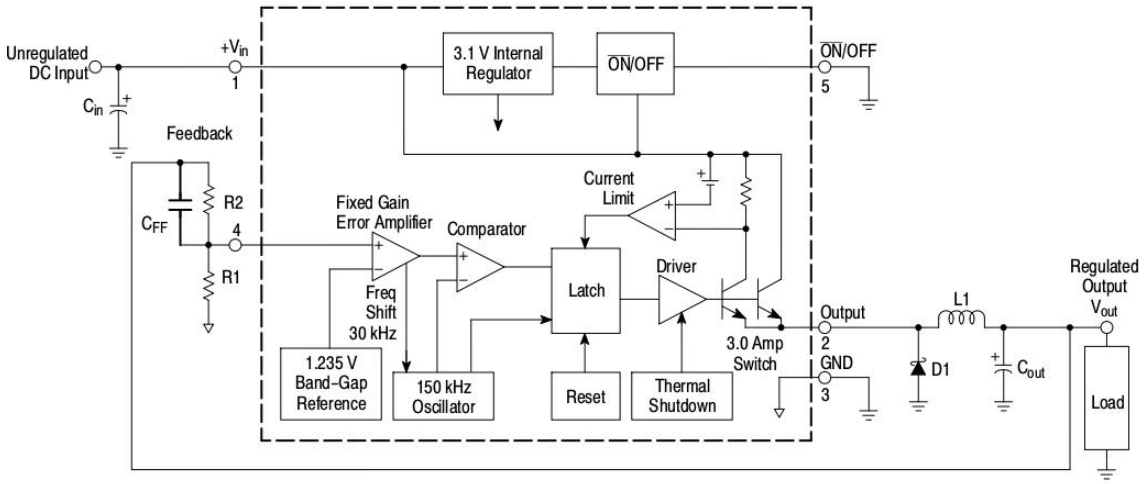


Figure 1. Typical Application and Internal Block Diagram

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Maximum Supply Voltage	V_{in}	45	V
$\overline{ON}$/OFF Pin Input Voltage	–	$-0.3\ V \le V \le +V_{in}$	V
Output Voltage to Ground (Steady−State)	–	−1.0	V
Power Dissipation			
Case 314B and 314D (TO−220, 5−Lead)	P_D	Internally Limited	W
Thermal Resistance, Junction−to−Ambient	$R_{\theta JA}$	65	°C/W
Thermal Resistance, Junction−to−Case	$R_{\theta JC}$	5.0	°C/W
Case 936A (D^2PAK)	P_D	Internally Limited	W
Thermal Resistance, Junction−to−Ambient	$R_{\theta JA}$	70	°C/W
Thermal Resistance, Junction−to−Case	$R_{\theta JC}$	5.0	°C/W
Storage Temperature Range	T_{stg}	−65 to +150	°C
Minimum ESD Rating (Human Body Model: C = 100 pF, R = 1.5 kΩ)	–	2.0	kV
Lead Temperature (Soldering, 10 seconds)	–	260	°C
Maximum Junction Temperature	T_J	150	°C

Stresses exceeding Maximum Ratings may damage the device. Maximum Ratings are stress ratings only. Functional operation above the Recommended Operating Conditions is not implied. Extended exposure to stresses above the Recommended Operating Conditions may affect device reliability.

PIN FUNCTION DESCRIPTION

Pin	Symbol	Description (Refer to Figure 1)
1	V_{in}	This pin is the positive input supply for the LM2596 step−down switching regulator. In order to minimize voltage transients and to supply the switching currents needed by the regulator, a suitable input bypass capacitor must be present (C_{in} in Figure 1).
2	Output	This is the emitter of the internal switch. The saturation voltage V_{sat} of this output switch is typically 1.5 V. It should be kept in mind that the PCB area connected to this pin should be kept to a minimum in order to minimize coupling to sensitive circuitry.
3	GND	Circuit ground pin. See the information about the printed circuit board layout.
4	Feedback	This pin is the direct input of the error amplifier and the resistor network R2, R1 is connected externally to allow programming of the output voltage.
5	$\overline{ON}$/OFF	It allows the switching regulator circuit to be shut down using logic level signals, thus dropping the total input supply current to approximately 80 μA. The threshold voltage is typically 1.6 V. Applying a voltage above this value (up to $+V_{in}$) shuts the regulator off. If the voltage applied to this pin is lower than 1.6 V or if this pin is left open, the regulator will be in the “on” condition.

OPERATING RATINGS (Operating Ratings indicate conditions for which the device is intended to be functional, but do not guarantee specific performance limits. For guaranteed specifications and test conditions, see the Electrical Characteristics.)

Rating	Symbol	Value	Unit
Operating Junction Temperature Range	T_J	−40 to +125	°C
Supply Voltage	V_{in}	4.5 to 40	V

SYSTEM PARAMETERS

ELECTRICAL CHARACTERISTICS Specifications with standard type face are for T_J = 25°C, and those with boldface type apply over full Operating Temperature Range −40°C to +125°C

Characteristics	Symbol	Min	Typ	Max	Unit
LM2596 (Note 1, Test Circuit Figure 15)					
Feedback Voltage (V_{in} = 12 V, I_{Load} = 0.5 A, V_{out} = 5.0 V,)	V_{FB_nom}		1.23		V
Feedback Voltage (8.5 V ≤ V_{in} ≤ 40 V, 0.5 A ≤ I_{Load} ≤ 3.0 A, V_{out} = 5.0 V)	V_{FB}	1.193 **1.18**		1.267 **1.28**	V
Efficiency (V_{in} = 12 V, I_{Load} = 3.0 A, V_{out} = 5.0 V)	η	**–**	73	–	%
Characteristics	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
Feedback Bias Current (V_{out} = 5.0 V)	I_b		25	100 **200**	nA
Oscillator Frequency (Note 2)	f_{osc}	135 **120**	150	165 **180**	kHz
Saturation Voltage (I_{out} = 3.0 A, Notes 3 and 4)	V_{sat}		1.5	1.8 **2.0**	V
Max Duty Cycle “ON” (Note 4)	DC		95		%
Current Limit (Peak Current, Notes 2 and 3)	I_{CL}	4.2 **3.5**	5.6	6.9 **7.5**	A
Output Leakage Current (Notes 5 and 6) Output = 0 V Output = −1.0 V	I_L		 0.5 6.0	 2.0 20	mA
Quiescent Current (Note 5)	I_Q		5.0	10	mA
Standby Quiescent Current ($\overline{ON}$/OFF Pin = 5.0 V (“OFF”)) (Note 6)	I_{stby}		80	200 **250**	μA
$\overline{ON}$/OFF PIN LOGIC INPUT					
Threshold Voltage			1.6		V
V_{out} = 0 V (Regulator OFF)	V_{IH}	2.2 **2.4**			V
V_{out} = Nominal Output Voltage (Regulator ON)	V_{IL}			1.0 **0.8**	V
$\overline{ON}$/OFF Pin Input Current					
$\overline{ON}$/OFF Pin = 5.0 V (Regulator OFF)	I_{IH}	–	15	30	μA
$\overline{ON}$/OFF Pin = 0 V (regulator ON)	I_{IL}	–	0.01	5.0	μA

1. External components such as the catch diode, inductor, input and output capacitors can affect switching regulator system performance. When the LM2596 is used as shown in the Figure 15 test circuit, system performance will be as shown in system parameters section.
2. The oscillator frequency reduces to approximately 30 kHz in the event of an output short or an overload which causes the regulated output voltage to drop approximately 40% from the nominal output voltage. This self protection feature lowers the average dissipation of the IC by lowering the minimum duty cycle from 5% down to approximately 2%.
3. No diode, inductor or capacitor connected to output (Pin 2) sourcing the current.
4. Feedback (Pin 4) removed from output and connected to 0 V.
5. Feedback (Pin 4) removed from output and connected to +12 V to force the output transistor “off”.
6. V_{in} = 40 V.

LM2596

TYPICAL PERFORMANCE CHARACTERISTICS (Circuit of Figure 15)

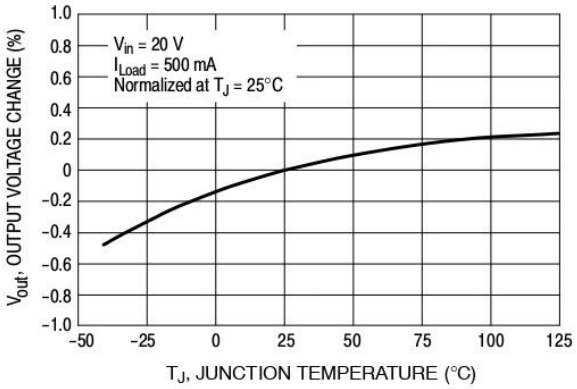


Figure 2. Normalized Output Voltage

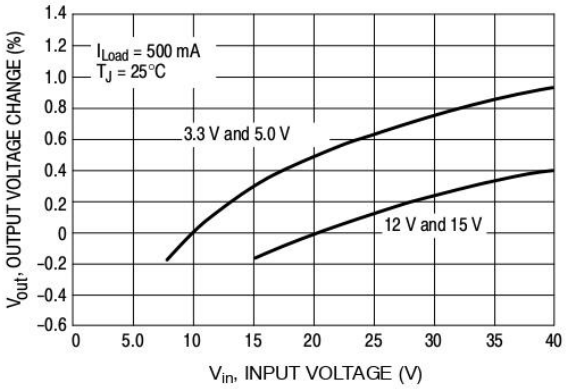


Figure 3. Line Regulation

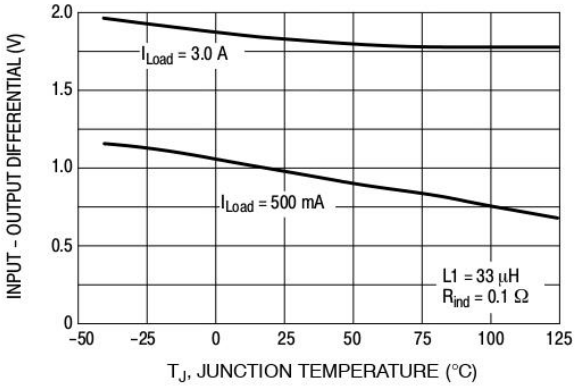


Figure 4. Dropout Voltage

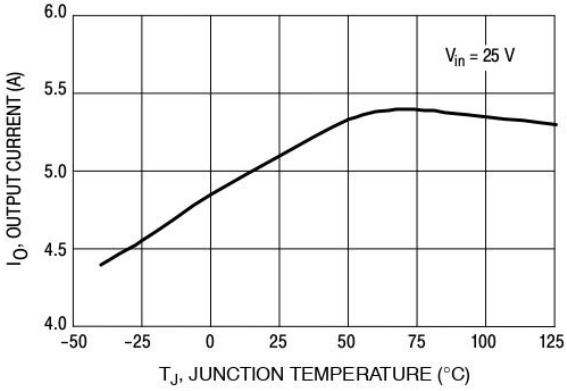


Figure 5. Current Limit

Fuente: (Onsemi, 2019)

ANEXO 10

DATASHEET POWER HD

Famous Professional Servo R&D Manufacturer
HuiDa RC International INC.
Address: 1707,Huashang Building,Xiapu district,Huizhou,Guangdong,China
E-mail:info@chd.hk Tel:86 752 2118844 Fax:86 752 2118860 WWW.CHD.HK

1. 使 用 環 境 條 件
Apply Environmental Condition：

No.	项目 item	规格 standard
1-1	保存溫度 Storage Temperature Range	-20℃～60℃
1-2	操作溫度 Operating Temperature Range	-10℃～50℃
1-3	操作電壓 Operating Voltage Range	4.8V~6.0V

2. 測 試 環 境
Standard Test Environment：

2-1	測試環境 Standard Test Environment	每一个检查必须是正常的温度和湿度进行测量，温度 25±5℃，相对湿度 65±10%,在按照本规范的标准测试条件下判断特征。 Every characteristic of the inspect must be normal temperature and humidity carry out the test , temperature 25±5℃ and relative humidity 65±10% of judgment made in accordance with this specification standard testing conditions.

3. 外 觀 檢 查
Appearance Inspection：

No.	项目 item	规格 standard
3-1	外觀尺寸 Outline Drawing	尺寸见附件 Dimension see the attachment
3-2	外觀 Appearance	无损坏，不允许影响功能 No damage which affects functions allowed

4. **電 氣 特 性**
Electrical Specification (Function of the Performance)：

No.	项目 item	4.8V	6.0V
4-1	空載轉速 Operating speed (at no load)	0.16 sec/60˚	0.14 sec/60˚
4-2	空載電流 Running current (at no load)	400 mA	500 mA
4-3	停止扭力 Stall torque (at locked)	8.6 kg-cm	9.8 kg-cm
4-4	停止電流 Stall current (at locked)	2300 mA	2500 mA
4-5	待機電流 Idle current (at stopped)	4 mA	5 mA

注：项目 4-2 定义平均值时，伺服器无负荷运行
Note: Item 4-2 definition is average value when the servo running with no load

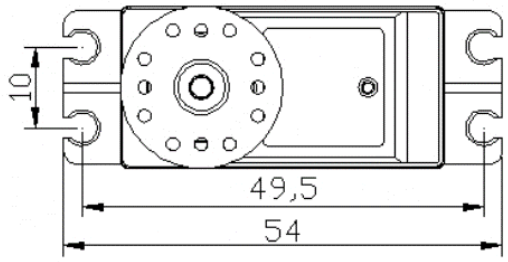
5. **機 械 特 性**
Mechanical Specification：

No.	项目 item	规格 standard
5-1	外觀尺寸 Overall Dimensions	见附件 See the drawing
5-2	機構極限角度 Limit angle	180˚ ± 10˚
5-3	重量 Weight	56 ± 1g
5-4	導線規格 Connector wire gauge	#28 PVC
5-5	導線長度 Connector wire length	300 ± 5 mm
5-6	舵片規格 Horn gear spline	25T/ ϕ 5.80
5-7	舵片種類 Horn type	十字，圆盘，星型，条型 Cross , Disk , Star, Double
5-8	減速比 Reduction ratio	1/479

6. 控 制 特 性

Control Specification：

No.	项目	规格
6-1	控制系統 Control system	改变脉冲宽度 Pulse Width Modification
6-2	放大器種類 Amplifier type	模拟控制器 Analog Controller
6-3	操作角度 Operating travel	90°（在 1000→2000 μsec）
6-4	中立位置 Neutral position	1500 μsec
6-5	脈波訊號虛位 Dead band width	4 μsec
6-6	旋轉方向 Rotating direction	顺时针（在 1500→2000 μsec） Counterclockwise (when1500→2000 μsec)
6-7	脈波寬度範圍 Pulse width range	800→2200 μsec
6-8	可作動角度範圍 Maximum travel	大约 165°（在 800→2200 μsec） Approx 165°(when800→2200 μsec)



Fuente: (Hobbyking - Servo)

ANEXO 11

DATASHEET SENSOR PING

PARALLAX

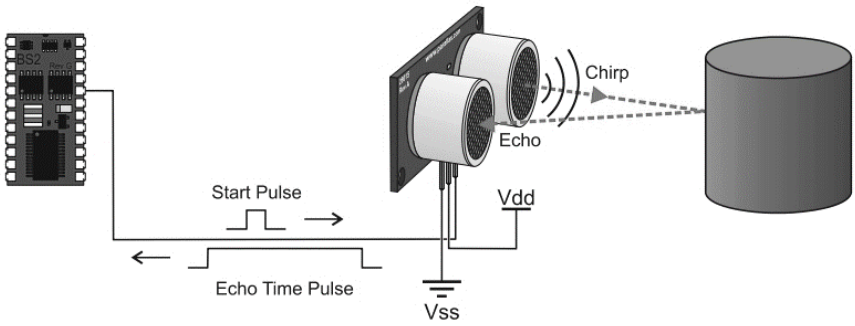
Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

PING))) Ultrasonic Distance Sensor (#28015)

The Parallax PING)))™ ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to microcontrollers such as the BASIC Stamp®, Propeller chip, or Arduino, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.



Features

- Range: 2 cm to 3 m (0.8 in to 3.3 yd)
- Burst indicator LED shows sensor activity
- Bidirectional TTL pulse interface on a single I/O pin can communicate with 5 V TTL or 3.3 V CMOS microcontrollers
- Input trigger: positive TTL pulse, 2 µs min, 5 µs typ.
- Echo pulse: positive TTL pulse, 115 µs minimum to 18.5 ms maximum.
- RoHS Compliant

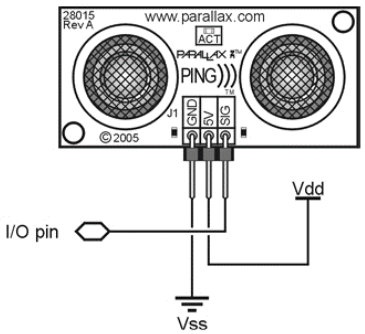
Key Specifications

- Supply voltage: +5 VDC
- Supply current: 30 mA typ; 35 mA max
- Communication: Positive TTL pulse
- Package: 3-pin SIP, 0.1" spacing (ground, power, signal)
- Operating temperature: 0 – 70° C.
- Size: 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)
- Weight: 9 g (0.32 oz)

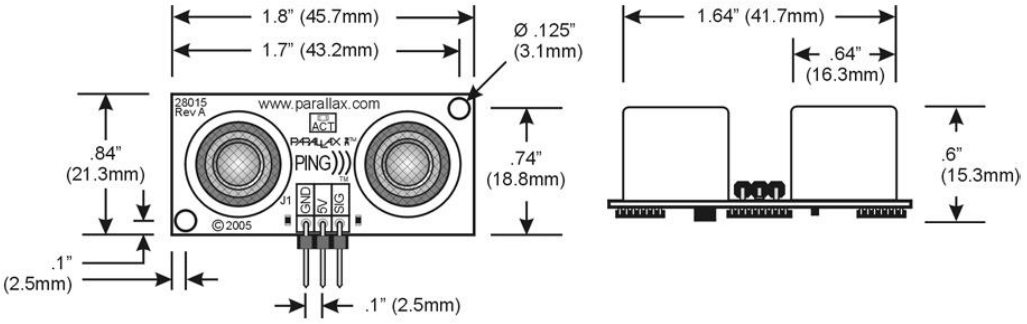
Pin Definitions

GND	Ground (Vss)
5 V	5 VDC (Vdd)
SIG	Signal (I/O pin)

The PING))) sensor has a male 3-pin header used to supply ground, power (+5 VDC) and signal. The header may be plugged into a directly into solderless breadboard, or into a standard 3-wire extension cable (Parallax part #800-00120).

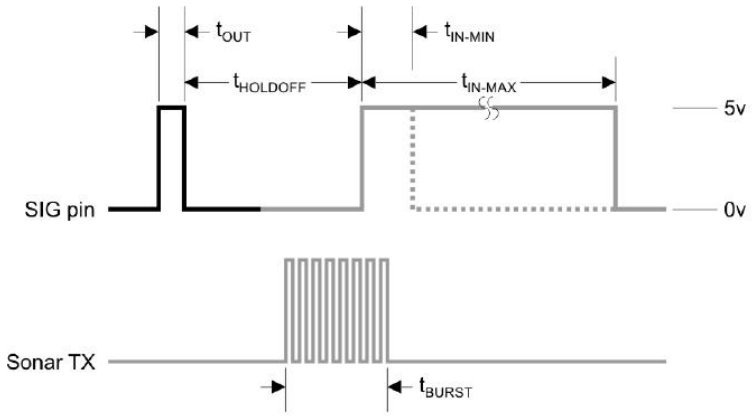


Dimensions



Communication Protocol

The PING))) sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.

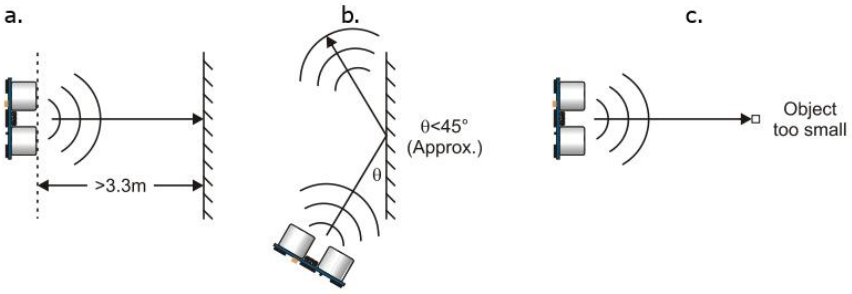


▬	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Return Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs

Practical Considerations for Use

Object Positioning

The PING))) sensor cannot accurately measure the distance to an object that: a) is more than 3 meters away, b) that has its reflective surface at a shallow angle so that sound will not be reflected back towards the sensor, or c) is too small to reflect enough sound back to the sensor. In addition, if your PING))) sensor is mounted low on your device, you may detect sound reflecting off of the floor.



Target Object Material

In addition, objects that absorb sound or have a soft or irregular surface, such as a stuffed animal, may not reflect enough sound to be detected accurately. The PING))) sensor will detect the surface of water, however it is not rated for outdoor use or continual use in a wet environment. Condensation on its transducers may affect performance and lifespan of the device.

Air Temperature

Temperature has an effect on the speed of sound in air that is measurable by the PING))) sensor. If the temperature (°C) is known, the formula is:

$$C_{air} = 331.5 + (0.6 \times T_C)\ m/s$$

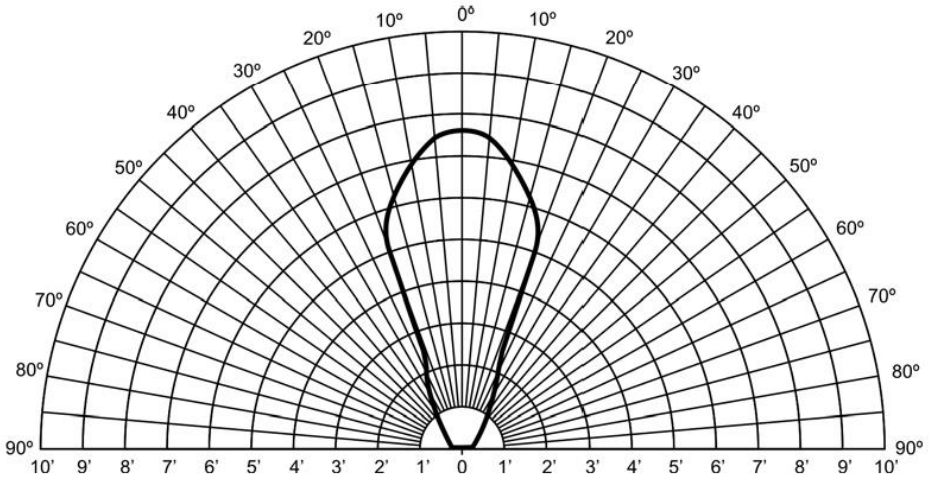
The percent error over the sensor's operating range of 0 to 70 ° C is significant, in the magnitude of 11 to 12 percent. The use of conversion constants to account for air temperature may be incorporated into your program (as is the case in the example BS2 program given in the Example Programs section below). Percent error and conversion constant calculations are introduced in Chapter 2 of *Smart Sensors and Applications*, a Stamps in Class text available for download from the 28029 product page at www.parallax.com.

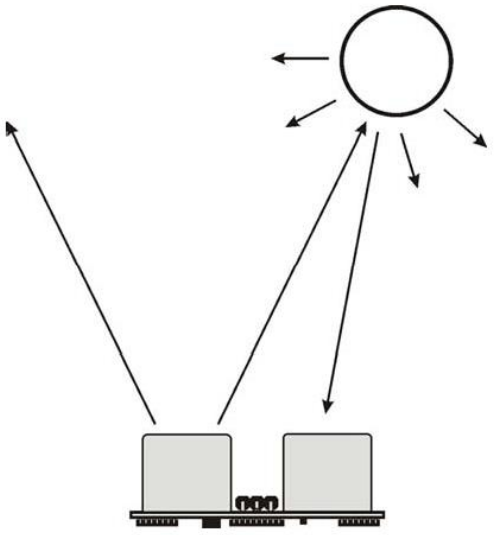
Test Data

The test data on the following pages is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

Test 1

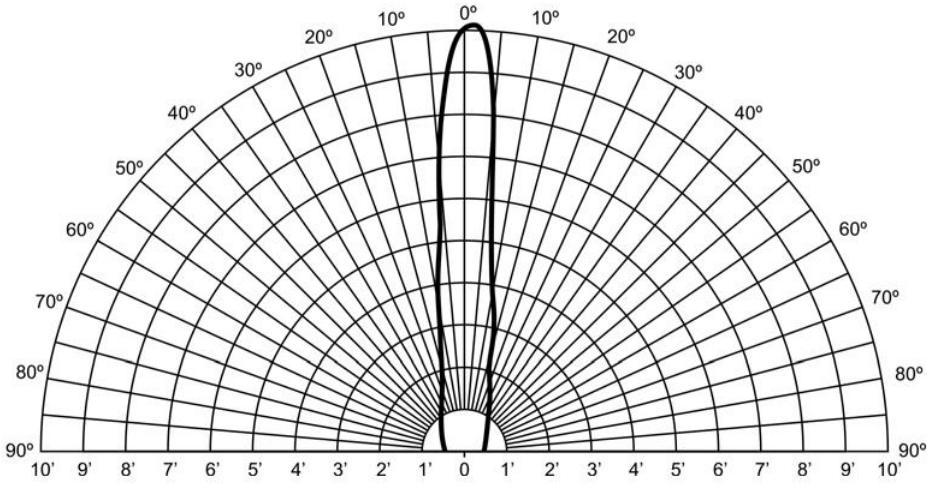
Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation

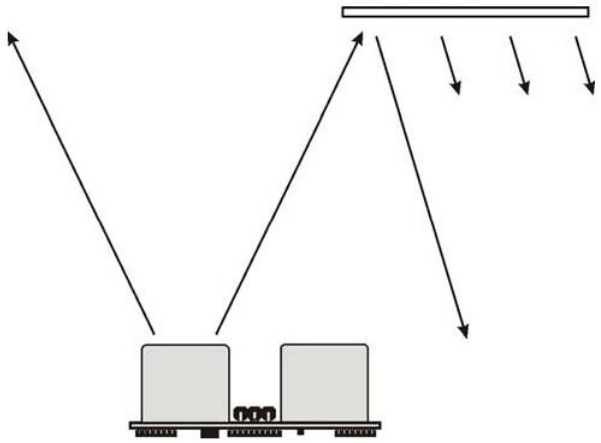




Test 2

Sensor Elevation: 40 in. (101.6 cm)
Target: 12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole
Target positioned parallel to backplane of sensor

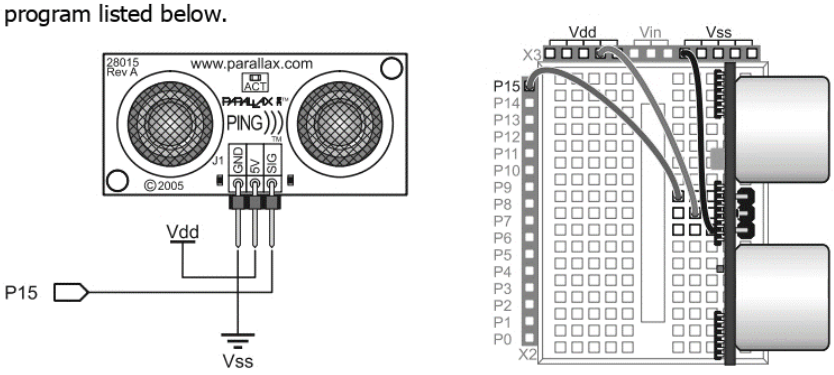




Example Programs

BASIC Stamp 2

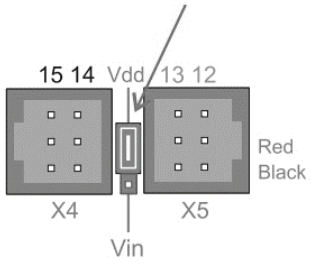
This circuit allows you to quickly connect your PING))) sensor to a BASIC Stamp® 2 via the Board of Education® breadboard area. The PING))) module's GND pin connects to Vss, the 5 V pin connects to Vdd, and the SIG pin connects to I/O pin P15. This circuit will work with the example BASIC Stamp program listed below.



Extension Cable and Port Cautions for the Board of Education

If you are connecting your PING))) sensor to a Board of Education platform using an extension cable, follow these steps:

1. When plugging the cable onto the PING))) sensor, connect Black to GND, Red to 5 V, and White to SIG.
2. Check to see if your Board of Education servo ports have a jumper, as shown at right.
3. If your Board of Education servo ports have a jumper, set it to Vdd as shown. Then plug the cable into the port, matching the wire color to the labels next to the port.
4. If your Board of Education servo ports do not have a jumper, do not use them with the PING))) sensor. These ports only provide Vin, not Vdd, and this may damage your PING))) sensor. Go to the next step.
5. Connect the cable directly to the breadboard with a 3-pin header as shown above. Then, use jumper wires to connect Black to Vss, Red to Vdd, and White to I/O pin P15.



Board of Education Servo Port Jumper, Set to Vdd

Fuente: (Parallax, 2018)