universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Trabajo de Fin de Grado

Diseño de un robot ‘Turtle’, con control remoto, implementando el lenguaje de programación LOGO

Design of a 'Turtle' robot, with remote control, implementing the LOGO programming language

Autor: Óscar Conde Pérez
Tutor: Fernando Jorge Fraile Fernández

(Julio, 2022)

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA EN ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA
Trabajo de Fin de Grado

ALUMNO: Óscar Conde Pérez

TUTOR: Fernando Jorge Fraile Fernández

TÍTULO: Diseño de un robot 'Turtle', con control remoto, implementando el lenguaje de programación LOGO

TITLE: Design of a 'Turtle' robot, with remote control, implementing the LOGO programming language

CONVOCATORIA: Julio, 2022

RESUMEN:
Desarrollo de un prototipo que dibuja formas geométricas regulares sobre una superficie plana, siguiendo las ordenes que un usuario le va dando con un sistema de comunicación inalámbrica. La idea nace del robot que crea a mediados de la década de los 70 el científico y matemático Seymour Papert en el MIT. Se trataba de un robot con forma de tortuga programado con el lenguaje Logo, que mediante ordenes como adelante, atrás, derecha o izquierda, iba dibujando formas sobre una superficie. El robot se creo para ayudar en la enseñanza de los niños en asignaturas como matemáticas, dibujo o informática. Con este prototipo se busca seguir con la idea del robot tortuga original, pero adaptándolo a las tecnologías actuales y actualizando el lenguaje de programación Logo a uno más actual y que pueda ser usado en los dispositivos actuales.
Para el control del prototipo se ha usado una placa Arduino, la cual permite controlar desde un mismo microcontrolador varios elementos y además son fáciles de programar. Para la comunicación inalámbrica se estudiaron diferentes alternativas para terminar escogiendo los módulos de comunicación Xbee. Las ordenes originales en el lenguaje Logo han sido adaptadas a Arduino incluyendo nuevas funciones para representar figuras geométricas predefinidas como cuadrado, triangulo, circulo o hexágono.

ABSTRACT:
Development of a prototype that draws regular geometric shapes on a flat surface, following the orders that a user gives it with a wireless communication system.
The idea stems from the robot created in the mid-1970s by the scientist and mathematician Seymour Papert at MIT. It was a turtle-shaped robot programmed with the Logo language, which, by means of commands such as forward, backward, right or left, drew shapes on a surface. The robot was created to help teach children in subjects such as mathematics, drawing and computer science.
The aim of this prototype is to continue with the idea of the original turtle robot, but adapting it to current technologies and updating the Logo programming language to a more modern one that can be used in current devices.
To control the prototype, an Arduino board has been used, which can control several elements from a single microcontroller and are also easy to program. For the wireless communication, different alternatives were studied to end up choosing the Xbee communication modules. The original commands in Logo language have been adapted to

Arduino including new functions to represent predefined geometric figures such as square, triangle, circle or hexagon.
Palabras clave: robot Turtle, comunicación inalámbrica, Xbee, Arduino

Firma del alumno:	**VºBº Tutor/es:**

Índices

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Memoria

1.1 OBJETO

El objetivo principal de este proyecto es el desarrollo de un prototipo, que mediante ordenes simples sea capaz de dibujar formas libres. Esta es una idea que nació ya en la década de los 70, cuando se creó un robot con forma de tortuga que mediante ordenes sencillas dibujaba formas geométricas. Esto se aplicó en la docencia para enseñar a los niños programación, geometría u otras disciplinas.

También este proyecto cuenta con varios objetivos secundarios o subobjetivos que se van a tener que abordar para conseguir un correcto funcionamiento final. Estos objetivos secundarios son:

- Estudiar diferentes formas de comunicación inalámbrica (Wifi, Bluetooth, por cable, Xbee…) y seleccionar la más apropiada para el proyecto.
- Estudiar los diferentes modelos de motores (motores DC, motores paso-a-paso…) para el movimiento del prototipo y cuales se adecuan más a nuestras necesidades.
- Adaptar las instrucciones originales de los años 70 a instrucciones que puedan ser usadas e interpretadas por lenguajes de programación modernos.
- Escoger el lenguaje de programación más idóneo en base al microprocesador que va a ser usado para el control del dispositivo.

1.2 ALCANCE

El fin de este proyecto es desarrollar un prototipo que sirva de apoyo pedagógico para los niños a la hora de dar sus primeros pasos en la programación, matemáticas, arte o geometría. También puede ser utilizado para enseñar formas geométricas u otras materias. El prototipo funciona moviéndose en línea recta, hacia adelante o hacia atrás, a derecha o izquierda un ángulo indicado. Además, posee un dispositivo capaz de subir o bajar un lapicero para pintar sobre una superficie plana las formas que le sean indicadas.

Para realizar el control del prototipo e indicarle hacia donde queremos que se desplace se han implementado una serie de instrucciones que a continuación van a ser explicadas:

- Adelante (): instrucción para desplazar el prototipo en línea recta hacia adelante.
- Retroceso (): instrucción para desplazar el prototipo en línea recta hacia atrás.
- Derecha (): instrucción para girar hacia la derecha.
- Izquierda (): instrucción para girar hacia la izquierda.

Estas son las instrucciones básicas para desplazar el robot. Dentro de los paréntesis se indicará la distancia en centímetros o el ángulo en grados centesimales.

Estas instrucciones básicas van acompañadas de otras dos más que permiten subir o bajar el lapicero para dibujar. Estas instrucciones son:

- Arriba (): instrucción para subir el lapicero.
- Abajo (): instrucción para bajar el lapicero.

Combinando estas instrucciones con las instrucciones básicas podemos dibujar cualquier forma que queramos. Por ejemplo: Si escribimos, *Abajo, Adelante (10), Izquierda (120), Adelante (10), Izquierda (120), Adelante (10), Arriba*; habremos dibujado un triángulo equilátero de 10cm de lado.

Así repitiendo instrucciones podemos dibujar cualquier forma que queramos. Pero para facilitar las cosas y que no sea tan engorroso dibujar polígonos de 5,6 o más lados, en los que hay que repetir muchas veces las instrucciones, se han creado funciones para que el prototipo haga estas figuras de forma directa:

- Cuadrado: instrucción para dibujar un polígono regular de 4 lados, preguntando al usuario la longitud del lado.

- Triangulo: instrucción para dibujar un polígono regular de 3 lados, preguntando al usuario la longitud del lado.
- Hexágono: instrucción para dibujar un polígono regular de 6 lados, preguntando al usuario la longitud del lado.
- Círculo: instrucción para dibujar una circunferencia. El círculo es aproximado mediante un polígono regular de 32 lados. En este caso se le pregunta al usuario el radio que desea de la circunferencia y el código hace el cálculo de cuanto debe de ser el lado del polígono.

De esta forma los niños o futuros usuarios finales poseen diversas herramientas para realizar el control del prototipo. Ya sea combinando instrucciones básicas para dibujar formas libres, pensando que instrucciones básicas se han de repetir para dibujar cierta forma geométrica o directamente indicando que forma dibujar al prototipo.

1.3 METODOLOGÍA

La realización de este proyecto ha sido estructurada en varias etapas para facilitar el desarrollo del mismo, buscando las mejores soluciones para cada una de las etapas planteadas y pudiendo así detectar posibles fallos o errores de funcionamiento antes de que afecten al desarrollo final del prototipo. Las etapas que se han planteado son:

- Elección del tipo de robot:

  Con las capacidades técnicas que necesitamos para nuestro proyecto (desplazarse representando formas en una superficie plana) existen diferentes tipos de máquinas capaces de llevarlas a cabo. Por lo que hubo que realizar un análisis de cada una de ellas, analizando sus pros y sus contras para acabar seleccionando la arquitectura más útil para nuestras necesidades.
  - Impresora 3D: gran sensibilidad de posicionamiento y no habría que preocuparse por el calibrado ya que el software de los motores lo realiza él solo antes de ponerse en funcionamiento. En su contra tiene su coste de montaje y mantenimiento y que, al no desplazarse, y nuestro proyecto está pensado para niños, resulta poco atractivo.
  - Máquinas CNC: son máquinas diferentes a las impresoras 3D, pero comparten la misma filosofía, realizar con un cabezal móvil una pieza procesada por ordenador. Comparten también los mismos defectos y virtudes de cara a nuestro proyecto. Además, incluyendo que las máquinas CNC pueden llegar a ser más peligrosas para los menores.

- Makeangelo: se trata de un robot capaz de hacer dibujos o murales[1]. Está constituido por un cabezal con un lapicero, sujeto a la estructura mediante un sistema de cuerdas y poleas movidas por unos motores que a su vez están controlados por un microcontrolador. Este tiene un código similar al de los softwares de las impresoras 3D que es capaz de interpretar una fotografía o dibujo para luego realizarlo. Es un sistema muy ingenioso con el que se consiguen muy buenos resultados. El problema es que funciona siempre en vertical, por lo que es necesario sujetarlo a la pared, hay que tener cuidado con las cuerdas a la hora de montarlo y de funcionar con él y necesita de una calibración antes de ponerse a funcionar. Además, no es muy interactivo.
- Robot car: se trata de un kit de montaje, muy económico, con un prototipo con la forma de un coche de 4 ruedas. De forma original el kit es para diseñar un vehículo de cuatro ruedas que mediante señales de infrarrojo sea capaz de esquivar obstáculos. Pero al ser un diseño que se puede modificar fácilmente hemos optado por esta opción, modificando el diseño a un vehículo de tres ruedas (como se explica en Análisis de soluciones) adaptándolo a nuestras necesidades de funcionamiento.

- Elección de la forma de comunicación:

  La primera característica a analizar era si la comunicación debía de ser inalámbrica o por cable. Si se realizaba la comunicación por cable, esta era directa y no hacía falta ningún dispositivo intermedio por lo que no se iban a producir interferencias. Pero en su contra, el rango de movimiento está limitado a la longitud del cable (unos 15 cm). Dado que es un proyecto pensado para ser usado en clases y manejado a distancia, este hándicap del movimiento eliminaba de forma automática la opción de comunicación por cable y hace que se escoja la comunicación inalámbrica.

  Sabiendo que no vamos a usar la comunicación por cable, era necesario saber qué forma de comunicación inalámbrica encajaba mejor en nuestro proyecto:

  - Bluetooth: fácil de implementar, pero podríamos tener problemas con las interferencias y con el rango de acción.
  - Wifi: fácil de implementar. Colocando la señal wifi en el canal adecuado se podría reducir mucho el riesgo de interferencias, pero el prototipo está pensado para ser usado en cualquier lugar y podremos encontrarnos lugares sin cobertura wifi, por lo que no se podría usar el prototipo.

- Xbee: ligeramente más complejo de implementar, pero reduce mucho el riesgo de las interferencias. Tiene un gran rango de alcance ya que él mismo crea su red de comunicación. Por ello, ésta ha sido la forma de comunicación elegida para el proyecto. Más adelante, en el apartado Análisis de soluciones se explica más en profundidad esta tecnología.

- Elección del microcontrolador:

Sabiendo el tipo de robot que vamos a utilizar y la forma en que nos íbamos a comunicar era necesario escoger qué tecnología íbamos a usar para controlar el movimiento del prototipo. Existían dos opciones reales: usar RaspBerry Pi o Arduino.

- RaspBerry Pi: son ordenadores de bajo coste y forma compacta orientados a ser usados para el desarrollo de códigos o ser utilizados en prototipos. La ventaja de estas placas es su capacidad de procesamiento de datos y fácil adaptación a cualquier tecnología. A pesar de que existen diferentes modelos, incluso los más pequeños son grandes para ser colocados en el prototipo.

- Arduino: es un microcontrolador programable que permite controlar y alimentar otros dispositivos. Es muy sencillo de usar y de programar. Existen diferentes modelos de varios tamaños los cuales sí encajan bien dentro del prototipo.

Ambas alternativas son muy atractivas para ser usadas en el prototipo, pero se decidió de usar el Arduino, modelo UNO, debido a tener menor coste, mejor adaptación con la comunicación Xbee y tener experiencia previa usándolo.

Una última etapa sería escoger el lenguaje de programación más adecuado, pero al haber sido escogido un microcontrolador Arduino, el lenguaje de programación nos viene ya dado por él. Usando las herramientas que ya facilita Arduino para la programación y uso de sus placas. [2]

Una vez analizadas las diferentes alternativas que existen a la hora de desarrollar el prototipo, se pasó a la fase de montaje y desarrollo del prototipo.

- Fase de montaje

En esta fase se va a probar que todas las decisiones que hemos tomado previamente son las correctas y funcionan según lo esperado. También esta fase sirve para probar los diferentes actuadores y seleccionar los que proporcionan un mejor funcionamiento. Esto se analiza en el apartado Elección de arquitectura del sistema.

- Fase final

Con el prototipo ya montado, se le realizan pruebas de funcionamiento a distintos niveles: se pone a prueba la comunicación, se sube y se baja el lapicero, se prueba el giro de las ruedas… Para corregir posibles fallos y determinar ya el montaje final del prototipo.

1.4 ANTECEDENTES

La preocupación por la enseñanza infantil ha sido una constante a lo largo de la historia. Se ha buscado siempre que los conocimientos que se les imparten a los jóvenes estén actualizados a los tiempos en los que viven. Para ello es necesario ofrecer la mejor formación posible a los profesores y proporcionarles a estos las herramientas necesarias para que puedan dar sus lecciones de la mejor de las maneras.

Los acontecimientos históricos que se produjeron durante el inicio del s.XX (Gran Depresión, Primera Guerra Mundial, Segunda Guerra Mundial, Guerra Fría…) provocaron un aumento de la inversión en investigación tecnológica y a un descubrimiento y desarrollo de nuevas tecnologías.

Con el final de estos sucesos históricos muchos de los avances tecnológicos que se habían logrado fueron aplicados en la vida de la población civil. Así en a partir de los años 50 muchas casas pudieron empezar a tener dispositivos tecnológicos tales como televisiones, frigoríficos, microondas, etc.

La sociedad se había tecnologizado, pero los métodos educativos que se seguían utilizando en las escuelas eran los propios de años atrás. No estaban adecuados a los tiempos.

En la década de los años 60 Seymour Papert [3] llega a los Estados Unidos con la idea de introducir el uso de los ordenadores en las aulas como un método educativo que ayudaría a mejorar la creatividad, la innovación y el pensamiento computacional en los más jóvenes.

Seymour Papert [4][5] nació en 1928 en Pretoria, Sudáfrica, se gradúo en filosofía en la Universidad de Witwatersrand en 1949 y en 1952 obtendría el doctorado en Matemáticas por la misma universidad.

Se destacó como un activista contra el apartheid en sus años universitarios. Trabajó en la Universidad de Ginebra como discípulo de Jean Piaget [6], reputado psicólogo suizo, denominado por muchos como "el padre de la epistemología genética".

Piaget desarrollo la teoría constructivista del aprendizaje [7] [8] que argumenta que el aprendizaje es un proceso interno en el cual son las vivencias con nuestro entorno y las relaciones sociales las que permiten desarrollar la inteligencia individual de los niños. Esta teoría marco fuertemente a Papert.

En 1963, Seymour Papert ingresa en el Instituto Tecnológico de Massachusetts (de aquí en adelante MIT por su sigla en inglés) como investigador asociado.

En 1967 es nombrado codirector del Laboratorio de Inteligencia Artificial junto al profesor Marvin Minsky, que es considerado como uno de los creadores de la Inteligencia Artificial. De su trabajo conjunto al que se unieron años más tarde el expresidente del MIT Jerome Wiesner y el profesor Nicholas Negroponte, nace en 1985 el MIT Media Lab, considerado uno de los mejores laboratorios de robótica del mundo. La creación de este nuevo laboratorio fue patrocinada (y sigue siéndolo a día de hoy) por la marca de bloques de construcción LEGO (más adelante se explica con más atención esta colaboración de la marca con el laboratorio).

Con el paso de los años, Papert desarrolla su propia visión de la teoría de Piaget, desarrollando lo que él mismo denomino cómo construccionismo, que predica que, en la relación con el medio, cada individuo puede crear una forma de pensamiento diferente al resto, es decir, una forma de pensar personal. En su libro *Constructionism* [9] define a través del siguiente ejemplo este nuevo modelo educativo: *"La línea de descendencia directa del construccionismo desde el modelo de la escultura de jabón es claramente visible. La definición más simple de construccionismo evoca la idea de aprender haciendo y esto es lo que estaba ocurriendo cuando los estudiantes trabajaban en sus esculturas de jabón".* (S. Papert, I. Harel , 1991, Constructionism p. 13)

En el año 1967, como forma de desarrollar la inteligencia individual y como medio para tener una herramienta para forzar el aprendizaje mediante la experiencia, Seymour Papert junto a Wallace Feurzeig, jefe de equipo de desarrollo de la compañía Bolt Beranek & Newman (BBN technologies) y la científica computacional especializada en inteligencia artificial Cynthia Solomon crean la primera versión de LOGO. [10][11]

Se trataba de un primer programa que representaba en la pantalla del ordenador una tortuga, haciendo las veces de cursor, que mediante un conjunto de órdenes muy sencillas como son adelante, atrás, derecha e izquierda, permitían a los niños generar gráficos con los que aprender geometría,

matemáticas, programación, arte…Era una herramienta que le permitía llevar a la práctica su teoría construccionista del aprendizaje.

El lenguaje de programación LOGO deriva de LISP (List Processor), basado en la modularidad, la extensión, la interactividad y la flexibilidad.

El nombre de LOGO fue escogido por Feurzeig y no se trata de ningún tipo de acrónimo, sino que deriva del griego y significa "pensamiento".

Los colegios en las décadas de los 60 y 70 no contaban con el material informático necesario para que LOGO fuera accesible en las clases. Por lo que se llevó de la pantalla a la vida real a la tortuga. Se diseñó un robot con la forma de este animal que se colocaba sobre una superficie plana y se movía siguiendo las instrucciones que le programaban los estudiantes.

Siguiendo las ideas de su teoría construccionista, Seymour Papert publica en 1980 su libro más importante titulado *Mindstorms: Children, Computers and Powerful Ideas* [12] en el que argumenta en contra de las ideas que buscan usar los ordenadores para programar a los niños y defiende la programación de los ordenadores por parte de los niños ya que al hacerlo se establece una sensación de dominio sobre una máquina moderna y potente y se desarrollan habilidades matemáticas, artísticas o del lenguaje. Es decir, para Papert no se trataba solo de programar sino de aprender mediante la programación otras disciplinas. Este concepto se explica mediante el ejemplo que se narra en el ensayo del primer capítulo del libro Constructionism de Seymour Papert e Idit Harel (Ablex Publishing Corporation, 1991[13]): Un alumno había mostrado un diseño de gráficos de pantalla que había programado usando el lenguaje LOGO. Al ser preguntado sobre cómo lo hizo, el alumno argumentó que tuvo que calcular ángulos y curvaturas para obtener las mejores formas. De esta forma el alumno, mediante la experiencia de una actividad, profundizó e interiorizó procesos matemáticos casi sin ser consciente.

Durante esta década de los 80, al tener la población mayor acceso a los ordenadores, se produjo una expansión a nivel mundial del uso de LOGO favorecida también por la traducción del lenguaje a diferentes idiomas. Además, fue incorporado por Apple para todos aquellos ordenadores que eran usados en las escuelas.

En 1985 se presentó LOGOWriter que tenía la capacidad de procesar texto, poseía una interfaz simplificada haciéndola más intuitiva y fácil de entender. La capacidad de procesar texto venía implementada en varios idiomas lo que hizo que esta aplicación fuera popular en todo el mundo.

Durante esta época se produjo el trabajo conjunto entre Seymour Papert, Mitchel Resnick y la compañía LEGO creando LEGO LOGO desarrollando un sistema que permitía conectar actuadores (motores, luces, sensores…) a una estructura construida con bloques de ladrillos de LEGO. El primer juguete robótico creado a partir de esta unión se llamó LEGO TC LOGO y permitía programar con lenguaje LOGO un juguete diseñado por uno mismo con los ladrillos de juguete. El último modelo es el que se utiliza a día de hoy y se llama LEGO Mindstorms (en homenaje al título del libro de Papert).

Mitchel Resnick nació en 1956 en Estados Unidos. Estudió en física en la Universidad de Princeton (1978) y un máster y un doctorado en informática en el MIT (1988, 1992). Seguidor de las teorías de Papert, comienzan a trabajar juntos en el grupo de epistemología y aprendizaje a partir de la creación del Media Lab del MIT en 1985.

Resnick es el creador del lenguaje de programación por bloques Scratch. Un lenguaje de programación visual, en el que combinando y encajando bloques se consigue crear un código. Es un lenguaje que se ha extendido rápidamente por todas los colegios e institutos para enseñar a programar a los alumnos ya que, aparte de ser muy visual e intuitivo, posee una curva de aprendizaje muy acelerada.

Scratch es una herramienta que forma parte de lo que Resnick denomina como “espiral de pensamiento creativo”. Esta “espiral” se inicia con un niño imaginando qué es lo que quiere hacer, pasa a la fase de crear mediante LEGO-LOGO o programando en Scratch. Una vez que lo tiene creado puede jugar con él y compartirlo con más gente. En ese proceso puede reflexionar sobre lo que ha creado e imaginar nuevas cosas, comenzando otra vez la espiral.

1.5 NORMAS Y REFERENCIAS

1.5.1 BIBLIOGRAFÍA

[1] "Makelangelo 5 – Marginally Clever Robots." [Online]. Available: https://www.marginallyclever.com/products/makelangelo-5/. [Accessed: 09-Jun-2022].

[2] B. Evans, "*Beginning Arduino Programming"*, 1ª ed, Apress Berkeley, CA, Technology in action, 2011.

[3] "Gracias Seymour Papert y Mitchel Resnick por cambiar el mundo." [Online]. Available: https://juegosrobotica.es/gracias-papert-resnick/#. [Accessed: 31-May-2022].

[4] "Profesor Seymour Papert." [Online]. Available: http://www.papert.org/. [Accessed: 31-May-2022].

[5] "En memoria: Seymour Papert — MIT Media Lab." [Online]. Available: https://www.media.mit.edu/posts/in-memory-seymour-papert/. [Accessed: 31-May-2022].

[6] "Biografia de Jean Piaget." [Online]. Available: https://www.biografiasyvidas.com/biografia/p/piaget.htm. [Accessed: 09-Jun-2022].

[7] M. P. J. Saldarriaga-zambrano, M. G. R. Bravo-cedeño, and M. M. R. Loor-, "La Teoria Constructivista De Jean Piaget Y Su Significación" vol. 2, Universidad Laica Eloy Alfaro de Manabí, Manta Ecuador . pp. 127–137, 2016.

[8] D.G.Arévalo Maldonado y M.G. Ñauta Herrera, *"Estado Actual del desarrollo de destrezas lectoras en el cuarto año de educación básica de acuerdo a la teoría piagetana"*,Tesis previa a la obtención del título, Universidad de Cuenca, Cuenca, Ecuador, 2010-2011

[9] J Warren, J.Adams and H.Molle "*Arduino Robotics"*, 1ªed, Apress Berkeley, CA, Technology in action, 2011

[10] "Historia de Logo, el lenguaje para aprender a programar." [Online]. Available: https://hipertextual.com/2019/02/logo-tortuga-lenguaje-programacion. [Accessed: 01-Nov-2020].

[11] "Logo History." [Online]. Available: https://el.media.mit.edu/logo-foundation/what_is_logo/history.html. [Accessed: 08-Nov-2020].

[12] S. Papert, *"Mindstorms: children, computers, and powerful ideas."*,1ª ed, New York, Basic Books, Inc., Publishers, 1980.

[13] “Situating Constructionism.” [Online]. Available: http://www.papert.org/articles/SituatingConstructionism.html. [Accessed: 31-May-2022].

[14] “Aplicación web de AutoCAD: editor y visor CAD en línea | Autodesk.” [Online]. Available: https://web.autocad.com/acad/me. [Accessed: 15-Jun-2022].

[15] “Software | Arduino.” [Online]. Available: https://www.arduino.cc/en/software. [Accessed: 15-Jun-2022].

[16] “XCTU - Descarga e instalación de la plataforma de configuración para soluciones XBee/RF | Digi International.” [Online]. Available: https://es.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu. [Accessed: 15-Jun-2022].

[17] “BOE.es - BOE-A-2007-973 Real Decreto 1580/2006, de 22 de diciembre, por el que se regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos.” [Online]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2007-973. [Accessed: 22-Apr-2022].

[18] “BOE.es - DOUE-L-2014-80628 Directiva 2014/35/UE del Parlamento Europeo y del Consejo, de 26 de febrero de 2014, sobre la armonización de las legislaciones de los Estados miembros en materia de comercialización de material eléctrico destinado a utilizarse con determinados límites de tensión.” [Online]. Available: https://www.boe.es/buscar/doc.php?id=DOUE-L-2014-80628. [Accessed: 22-Sep-2021].

[19] “BOE.es - BOE-A-2011-14252 Real Decreto 1205/2011, de 26 de agosto, sobre la seguridad de los juguetes.” [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2011-14252&p=20201107&tn=0. [Accessed: 22-Apr-2022].

[20] “BOE.es - BOE-A-2011-14252 Real Decreto 1205/2011, de 26 de agosto, sobre la seguridad de los juguetes.” [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2011-14252&p=20130214&tn=2. [Accessed: 22-Sep-2021].

[21] “Programación con LOGO » Recursos educativos digitales.” [Online]. Available: https://www3.gobiernodecanarias.org/medusa/ecoescuela/recursosdigitales/2015/02/11/programacion-con-logo/. [Accessed: 15-Jun-2022].

[23] “¿Qué es XBee? XBee.cl - Comunicación Inalámbrica para Tus Proyectos.” [Online]. Available: https://xbee.cl/que-es-xbee/. [Accessed: 15-Jun-2022].

[24] “Arduino IDE en Windows Linux y Mac.” [Online]. Available:

https://programarfacil.com/blog/arduino-blog/arduino-ide/. [Accessed: 21-Jun-2022].

[25] Parlamento Europeo and Consejo de la Unión Europea, “Por la que se establece el Código Europeo de las Comunicaciones Electrónicas,” pp. 36–241, 2018.

[26] “ELEGOO Smart Robot Car Kit V3.0 Plus/V3.0/V2.0/V1.0 Tutorial – ELEGOO Official.” [Online]. Available: https://www.elegoo.com/blogs/arduino-projects/elegoo-smart-robot-car-kit-v3-0-plus-v3-0-v2-0-tutorial. [Accessed: 10-May-2022].

[27] Catálogo *"28BYJ-48 - 5V Stepper Motor"*, Kiatronics electronic design and manufacture.

[28] “Motor paso a paso 28BYJ-48 con Arduino y driver ULN2003.” [Online]. Available: https://www.luisllamas.es/motor-paso-paso-28byj-48-arduino-driver-uln2003/. [Accessed: 14-Jun-2022].

[29] Catálogo, *“Datasheet ULN2001A-ULN2002A-ULN2003A-ULN2004A”*STMicroelectronics no. February, pp. 1–8, 2000.

[30] “Controlar un servo con Arduino.” [Online]. Available: https://www.luisllamas.es/controlar-un-servo-con-arduino/. [Accessed: 15-Jun-2022].

[31] Catálogo, *“Sg90 9g Micro Servo”*, Micro Servo,Tech. Rep. 2015.

[32] “Controlar un servo con Arduino.” [Online]. Available: https://www.luisllamas.es/controlar-un-servo-con-arduino/. [Accessed: 31-Aug-2021].

[33] “Tipos de motores rotativos para proyectos de Arduino.” [Online]. Available: https://www.luisllamas.es/tipos-motores-rotativos-proyectos-arduino/. [Accessed: 22-Sep-2021].

[34] M. Etxebarria Isuskiza, *"Arduino: La tecnología al alcance de todos"*, 1ª ed, Madrid, Creaciones Copyright, S.L. 2013

[35] “Motores Arduino | Aprendiendo Arduino.” [Online]. Available: https://aprendiendoarduino.wordpress.com/2017/06/24/motores-arduino/. [Accessed: 15-Jun-2022].

[36] “Servo - Arduino Reference.” [Online]. Available: https://www.arduino.cc/reference/en/libraries/servo/. [Accessed: 15-Jun-2022].

[37] “Arduino - Math Library.” [Online]. Available: https://www.tutorialspoint.com/arduino/arduino_math_library.htm. [Accessed: 15-Jun-2022].

[38] “arduineando – Xbee y Arduino.” [Online]. Available: https://www.arduineando.com/xbee-y-arduino/. [Accessed: 29-Mar-2022].

[39] C. Bell, "*Beginning sensor networks with Arduino and Raspberry Pi"*, 2ª ed, Apress Berkeley, CA, Technology in action, 2020. DOI 10.1007/978-1-4302-5825-4

[40] “ZigBee/XBee | Aprendiendo Arduino.” [Online]. Available: https://aprendiendoarduino.wordpress.com/2016/11/16/zigbeexbee/. [Accessed: 29-Mar-2022].

[41] J. Martín moreno y D. Ruiz Fernández, *"Informe Técnico: Protocolo ZigBee (IEEE 802.15.4)"*, Junio de 2007.

[42] D. Garcia, *"El estándar IEEE 802.15.4"* Tech. Rep. Marzo 2018

[43] “ZigBee/XBee | Aprendiendo Arduino.” [Online]. Available: https://aprendiendoarduino.wordpress.com/2016/11/16/zigbeexbee/. [Accessed: 10-May-2022].

[44] Catálogo, *"Servo Motor SG90"*, Tower Pro. .

[45] “Leonardo | Arduino Documentation | Arduino Documentation.” [Online]. Available: https://docs.arduino.cc/hardware/leonardo. [Accessed: 15-Jun-2022].

[46] “Arduino - ArduinoBoardLeonardo.” [Online]. Available: https://www.arduino.cc/en/main/arduinoBoardLeonardo. [Accessed: 18-Apr-2022].

[47] “Legacy XBee S1 802.15.4 Product Datasheet | Digi International.” [Online]. Available: https://www.digi.com/resources/library/data-sheets/ds_xbeemultipointmodules. [Accessed: 11-Jul-2022].

[48] *"Xbee Shield V1.1"*, Itead Studio, Tech. sup. Agosto 2011

[49] España Ministerio de Ciencia y Tecnología and Agencia Estatal Boletín Oficial del Estado (España), *Reglamento electrotécnico para baja tensión e ITC.* 2019.

[50] I. Harel and S. Papert, *"Constructionism"*, 1ªed, Norwood, New Jersey, Ablex Publishing Corporation, 1991

1.5.2 PROGRAMAS DE CÁLCULO, DISEÑO Y SIMULACIÓN

[14] "Aplicación web de AutoCAD: editor y visor CAD en línea | Autodesk." [Online]. Available: https://web.autocad.com/acad/me.

[15] "Software | Arduino." [Online]. Available: https://www.arduino.cc/en/software.

[16] "XCTU - Descarga e instalación de la plataforma de configuración para soluciones XBee/RF | Digi International." [Online]. Available: https://es.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu.

1.5.3 DISPOSICIONES LEGALES Y NORMATIVA

[17] "BOE.es - BOE-A-2007-973 Real Decreto 1580/2006, de 22 de diciembre, por el que se regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos." [Online]. Available: https://www.boe.es/buscar/doc.php?id=BOE-A-2007-973. [Accessed: 22-Apr-2022].

[18] "BOE.es - DOUE-L-2014-80628 Directiva 2014/35/UE del Parlamento Europeo y del Consejo, de 26 de febrero de 2014, sobre la armonización de las legislaciones de los Estados miembros en materia de comercialización de material eléctrico destinado a utilizarse con determinados límites de tensión." [Online]. Available: https://www.boe.es/buscar/doc.php?id=DOUE-L-2014-80628. [Accessed: 22-Sep-2021].

[19] "BOE.es - BOE-A-2011-14252 Real Decreto 1205/2011, de 26 de agosto, sobre la seguridad de los juguetes." [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2011-14252&p=20201107&tn=0. [Accessed: 22-Apr-2022].

[20] "BOE.es - BOE-A-2011-14252 Real Decreto 1205/2011, de 26 de agosto, sobre la seguridad de los juguetes." [Online]. Available: https://www.boe.es/buscar/act.php?id=BOE-A-2011-14252&p=20130214&tn=2. [Accessed: 22-Sep-2021].

1.6DEFINICIONES Y ABREVIATURAS

Logo: lenguaje de programación de alto nivel pensado para ser usado por niños por ser fácil de aprender ya que está basado en una serie de instrucciones muy sencillas. [21]

MIT (Massachusetts Institute of Technology): universidad privada localizada en Cambridge, Massachusetts (Estaos Unidos). Se considera una de las universidades más prestigiosas en innovación e investigación tecnológica del mundo. [22]

Xbee: dispositivos electrónicos integrados que permiten el establecimiento de una comunicación inalámbrica entre dispositivos mediante el protocolo de comunicación de red IEEE 802.15.4[23]

PLA (ácido poliláctico o poliácido láctico): es un polímero sintético derivado de materias primas naturales como el almidón de maíz o de yuca. Se usa como material de los filamentos de las impresoras 3D dadas sus buenas características de resistencia y durabilidad.

XCTU: aplicación multiplataforma y gratuita creada para facilitar el control, configuración y prueba de los módulos Xbee por medio de un entorno gráfico fácil de utilizar. [16]

IDE: aplicación multimedia que sirve como entorno de desarrollo integrado para Arduino. Permite la programación hardware de la gran parte de las placas Arduino que hay en el mercado. [25]

1.7 REQUISITOS DE DISEÑO

1.7.1 Comunicación

Las comunicaciones inalámbricas han de tener en cuenta la directiva (UE) 2018/1972 del parlamento europeo por el que se establece el Código Europeo de las Comunicaciones Electrónicas.

Además, se ha de tener en cuenta que el mayor uso del prototipo será en zonas interiores y con afluencia de público por lo que las interferencias y el rango de alcance de la señal es algo a tener muy en cuenta a la hora de seleccionar el tipo de comunicación inalámbrica que se vaya a usar.

1.7.2 Control de posición y giro

Al tratarse de un prototipo de dibujo se busca que sea lo más preciso y exacto posible. Para ello es necesario tener un buen control sobre la posición y velocidad de giro de los motores encargados del desplazamiento del prototipo. Desechando aquellos motores que nos proporcionan aceleración y velocidad por aquellos en los que podamos tener un gran control de su posición, aunque sea necesario sacrificar la velocidad de giro.

1.8ANÁLISIS DE SOLUCIONES

1.8.1 Elección arquitectura del sistema

Para la ejecución de este prototipo se ha utilizado como base el Robot Car Kit SMARTV2.0 de Elegoo [26]. Utilizando varios de sus componentes como son las ruedas, las dos planchas que forman el cuerpo del robot y la caja de las pilas. El resto de piezas y componentes utilizados han sido estudiados y diseñados para acoplarse a las necesidades del diseño o del funcionamiento del prototipo.

La elección de la arquitectura final del sistema se puede desgranar en tres bloques:

- Elementos de sujeción.
- Posición
- Centro de giro

Elementos de Sujeción:

- Sujeción elemento de dibujo:

El primer problema que había que afrontar era el diseño de un sistema capaz de sujetar y mover en el eje vertical (arriba y abajo) un rotulador, lapicero, cera de color, etc. de diferentes longitudes y diámetros. El dispositivo diseñado para esta función consta de dos piezas. Una más pequeña y con forma de sombrero que se encarga de sujetar el rotulador con la ayuda de un tornillo, al estilo de cómo se sujeta un lápiz en un compás de dibujo. Además, esta pieza posee un alero (de ahí la similitud con la forma de un sombrero) para que el aspa del servomotor la haga subir o bajar. Los planos de estas piezas se encuentran en el apartado 3 Planos.

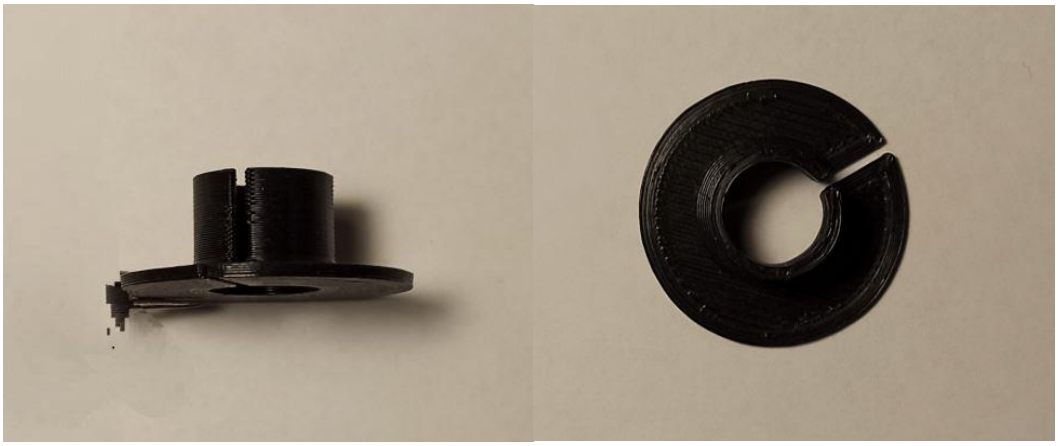


Figura 1 - 1 Sujeción lápiz

La segunda pieza es un cilindro en el que penetra, hasta que hace tope el “sombrero”, el rotulador y que lo mantiene en la vertical. Esta pieza también sirve de soporte para el servomotor que hace subir y bajar el rotulador o elemento de dibujo.

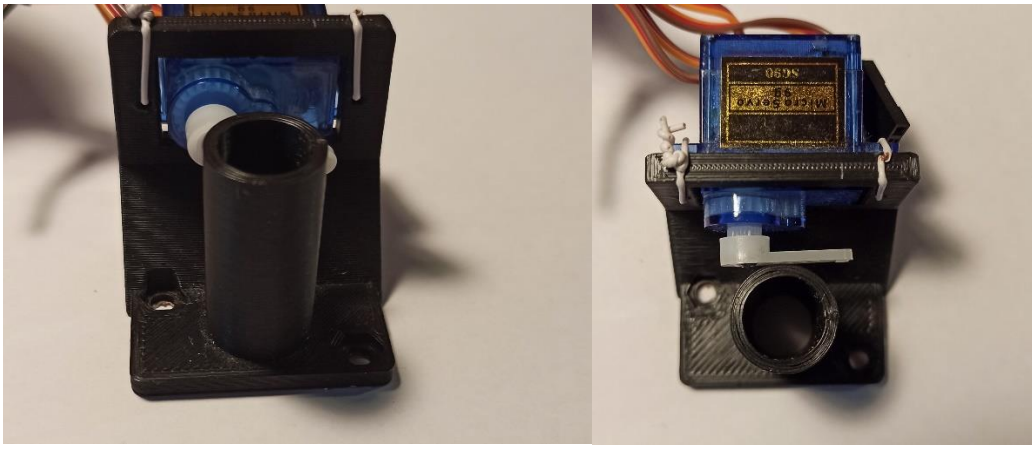


Figura 1 - 2 Elemento soporte de dibujo

Una vez se supo cómo íbamos a realizar la sujeción y movimiento del elemento de dibujo, faltaba resolver su colocación en el prototipo. En un inicio se colocó en la punta principal del robot. En esta zona gozaba de más espacio y era muy sencilla su instalación. Pero ya en las primeras pruebas se detectó un problema. Al encontrarse fuera del centro de giro del prototipo, al realizar un giro la punta del rotulador no permanecía fija, sino que realizaba un arco de circunferencia. Dando lugar a un funcionamiento erróneo.

Para solucionar este problema se decide que este conjunto de piezas este siempre colocado en el centro de giro del prototipo. (más adelante se explica la problemática con el giro y como esto provoca cambios en el centro de giro).

- Sujeción motores:

La sujeción de los motores originales no valía para los nuevos motores paso-a-paso. Se ha utilizado para su sujeción una pieza de diseño abierto diseñada específicamente para los motores 28BYJ-48 que son los utilizados.

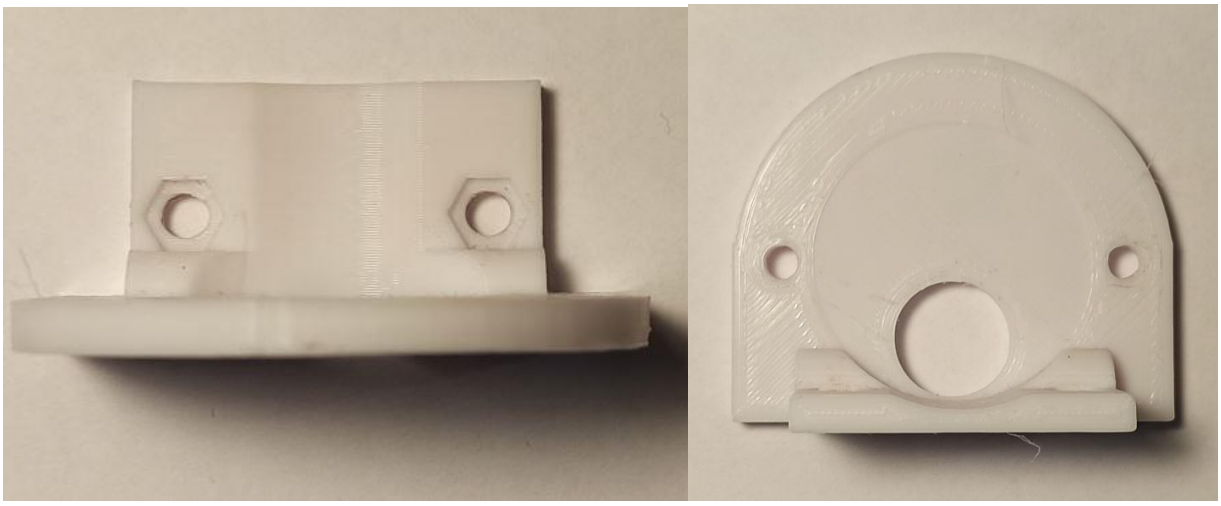


Figura 1 - 3 Soporte motores

- Sujeción ruedas

También había que acoplar las ruedas que teníamos al eje de los nuevos motores. Ya que este era más pequeño que el agujero que tienen las ruedas y estas bailaban al colocarse.

La pieza que se diseñó está compuesta por una espiga, que va encajada en el agujero de las ruedas y un agujero con la forma del eje del motor para que vaya encajado en este.

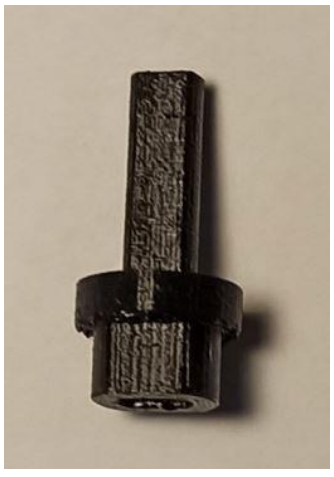


Figura 1 - 4 Pieza sujeción ruedas al motor

Posición

Al tratarse de un prototipo dedicado a dibujar formas geométricas era necesario que fuera muy preciso con los ángulos y puntos de giro.

En una fase inicial el robot estaba construido con motores geared down de corriente continua. Estos son muy fiables y fáciles de utilizar, pero a pesar de que otorgaban una buena velocidad de movimiento, no permitían tener un gran control sobre su posición. Lo que hizo descartar esta opción.

Descartados este tipo de motores se decidió probar con motores paso-a-paso. Estos no son tan sencillos de funcionar como los motores DC, pero permiten un control total de su posición y de su velocidad, que son las dos características principales que buscamos para este proyecto.

Se realizan pruebas con motores 28BYJ-48 [27] ya que son motores de pequeño tamaño, formados por cuatro bobinas colocadas en cruz, con una alimentación de 5V y un paso de 5.624 (64 pasos). Lo que nos permite tener una precisión en la posición del eje del motor de 0,088 grados.

Se decide utilizar estos motores dada su fiabilidad, tamaño, buen funcionamiento y el gran control de la posición que permiten.

Los motores girarán todos en el mismo sentido o no dependiendo del movimiento que queramos aplicar al prototipo:

- Adelante: todos los motores giran en sentido horario.
- Retroceso: todos los motores giran en sentido antihorario.
- Derecha: los motores del lado derecho giran en sentido antihorario y los del lado izquierdo en sentido horario.
- Izquierda: los motores del lado izquierdo giran en sentido antihorario y los del lado derecho en sentido horario.

Los motores 28BYJ-48 poseen 3 diferentes secuencias de movimiento [28]:

- Secuencia 1-Fase: se enciende una bobina de cada vez.

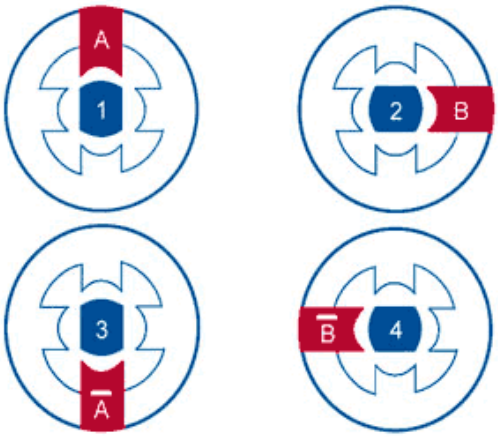


Figura 1 - 5 Secuencia encendido 1 fase (Fuente [28]

- Secuencia 2-Fases: en cada una de las cuatro fases del movimiento se van encendiendo dos bobinas consecutivas. Esto provoca un aumento del 41% del campo magnético por cada movimiento. Teniendo más fuerza para cada paso, pero con un peor consumo energético.

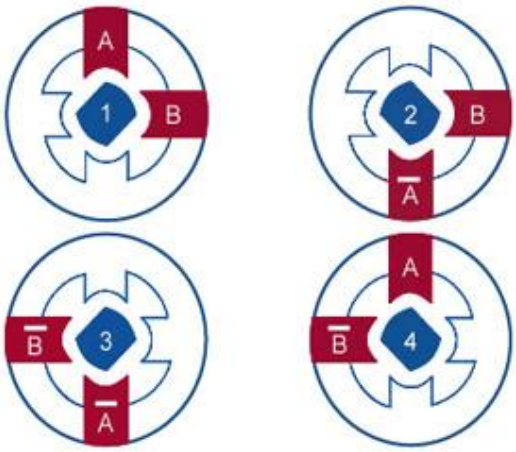


Figura 1 - 6 Secuencia encendido 2 fases (Fuente [28])

- Secuencia Medio Paso (half-step): El movimiento completo del motor se encuentra dividido en 8 fases. En las fases impares solo se enciende una bobina mientras que en las fases pares se produce el encendido de dos bobinas consecutivas. Al tener el doble de fases que la secuencia anterior, se consigue

el doble de exactitud en el movimiento. Además, se logra un ahorro de energía respecto a la secuencia de 2-Fases al no tener siempre dos bobinas activas.

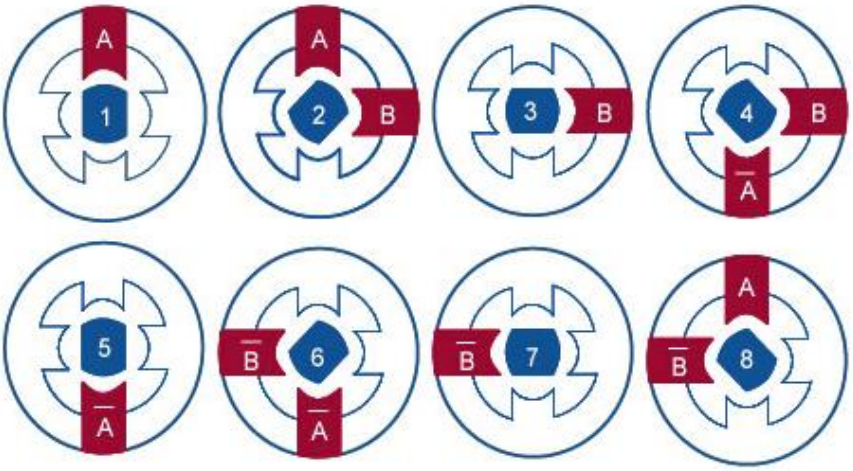


Figura 1 - 7 Secuencia encendido media fase (Fuente [28])

Analizando las diferentes secuencias de movimiento se optó por esta última, la secuencia de medio paso, al ser la que mejores características presentaba para el diseño del prototipo. La implementación de esta secuencia de movimiento se analizará más en profundidad en el apartado Software.

Por otro lado, para poder hacer funcionar estos motores desde un Arduino, es necesario colocar como etapa intermedia unos Drivers que aumenten la corriente que llega desde el sistema de control y que va hasta los motores. En este caso se utiliza el modelo ULN2003A [29]

En unas primeras pruebas con estos motores se colocaron los drivers, tal y como llega al realizar la compra, con cada motor. (Los drivers que vienen incluidos en la compra de los motores, vienen cada uno de ellos en una placa individual). Esto provoca una falta de espacio y una mala optimización de los recursos.

Los motores de un mismo lado del prototipo siempre van a girar en el mismo sentido (o bien en sentido horario o bien en sentido antihorario), por lo que pueden ser controlados con el mismo driver ambos motores. Por esta razón pasar de los 4 drivers iniciales a solo 2. Uno por cada par de motores de un mismo lado. (Esta es la arquitectura inicial del prototipo, más adelante se explica la arquitectura final y el porqué de esta, siendo necesarios solo 2 motores. Por tanto, al final sólo habrá un motor por cada driver).

Los drivers se sacaron de la placa comercial con la que llegan y se colocaron en una protoboard. Logrando así reducir el espacio que ocupan y poder tener todas las conexiones agrupadas en un mismo espacio. Esta placa de conexiones, además, permite reducir el número de conexiones directas que van al Arduino. Ya que las conexiones de GND y alimentación tanto de los motores, de los drivers y del servomotor de la pieza de sujeción del rotulador van todas ellas a esta placa. De esta forma solo la protoboard tiene una conexión directa con el Arduino (con dos entradas, la de alimentación a 5V y GND) y el resto de conexiones necesarias se realizan desde la protoboard.

Centro de giro

Para que el rotulador no realizara arcos de circunferencia durante el giro del prototipo y que se mantuviera fijo en un punto mientras este giraba, era necesario tener localizado el centro de giro. Este se encuentra siempre sobre los ejes de simetría.

En el primer montaje que se realizó del prototipo esto no se tuvo en consideración, y se colocó la pieza que sujeta al rotulador en el punto delantero del prototipo (como se explicó anteriormente en el punto de elemento sujeción de dibujo) provocando que el rotulador realizara arcos de circunferencia al girar.

Para subsanar este error, se colocó el rotulador en el centro de giro del prototipo. El diseño primero del prototipo era con 4 ruedas. Lo que producía 2 ejes de simetría. El eje que corta al prototipo en dos partes iguales verticalmente y el eje que lo corta en dos partes iguales horizontalmente.

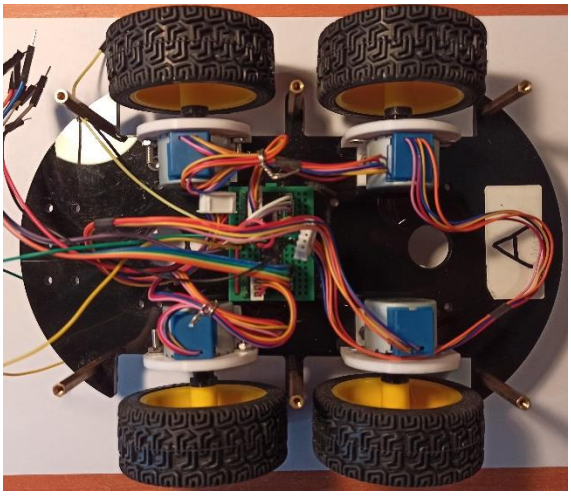


Figura 1 - 8 Montaje con 4 rueda

Una vez colocada ya la sujeción del rotulador en el centro de giro, se realizan nuevas pruebas de funcionamiento. El dibujo que realiza sale de forma continua sin arcos de circunferencia como era de esperar, pero el robot no gira de manera controlada. Las ruedas patinan o simplemente no giran cuando deberían de hacerlo.

Se realizan nuevas pruebas para buscar posibles soluciones:

- Se prueba el prototipo sobre diferentes superficies (sobre una mesa de madera, un folio blanco, el suelo de parqué y cerámico) para comprobar el agarre de las ruedas.
- Se baja la velocidad de giro de las ruedas hasta el punto de quedarse si fuerza para mover el prototipo
- Se aumenta la velocidad de giro de las ruedas hasta el punto de llegarse a salir alguna de ellas.
- Hacer que sólo giren las ruedas exteriores y las interiores permanezcan fijas.

Ninguna de estas posibles soluciones ayudó a solucionar el problema del giro. Por lo que se optó por pensar en un nuevo diseño.

Este nuevo diseño está inspirado en la distribución de las ruedas que tiene una carretilla elevadora de 3 ruedas. Las cuales están pensadas para girar y maniobrar en poco espacio. Características que buscamos para el prototipo.

Ahora, el nuevo diseño consta de 3 puntos de apoyo, 2 ruedas motrices y una rueda libre, en lugar de los 4 puntos de apoyo, 4 ruedas motrices, iniciales.

Las 2 ruedas motrices están colocadas en la parte delantera del prototipo y la rueda libre está colocada en el centro del prototipo a la altura de donde se encontraban las antiguas ruedas posteriores.

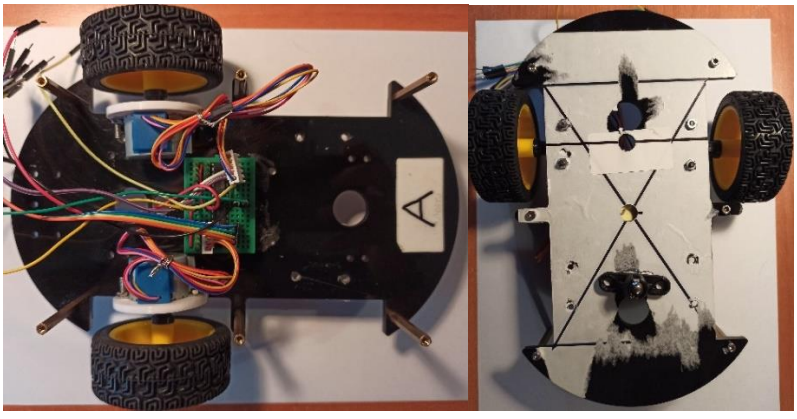


Figura 1 - 9 Montaje con 3 ruedas

La rueda libre está constituida por una bola metálica de 7 mm de diámetro sujeta por un soporte diseñado específicamente para ella.

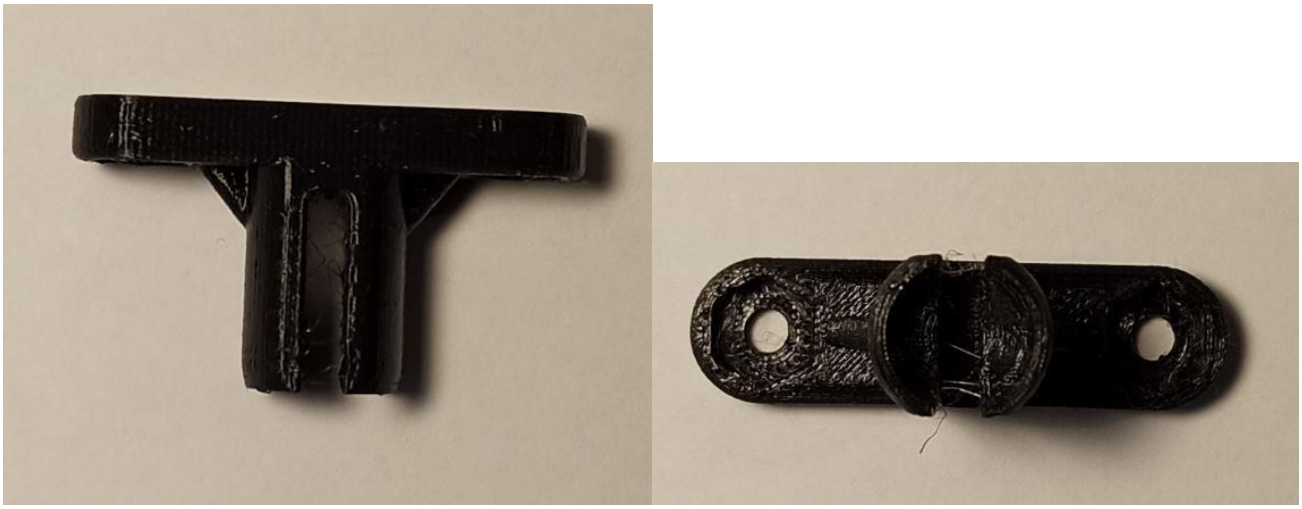


Figura 1 - 10 Soporte bola

Con esta nueva distribución el centro de giro cambia de posición también. El nuevo centro de giro se encuentra en el eje de simetría vertical justo donde corta con la línea que une las dos ruedas delanteras.

Por consiguiente, en este punto se coloca la pieza de sujeción del rotulador de dibujo y obliga a cambiar también la posición del Arduino a una posición más retrasada en el soporte del prototipo.

Con todos los cambios ya realizados, se efectuaron nuevas pruebas de funcionamiento y se confirma que se ha logrado solucionar la problemática del centro de giro, pero se detecta un nuevo problema. Al tener el rotulador solo un punto de sujeción y bastante elevado de la altura del papel, el rozamiento que se produce entre el papel y la punta del rotulador en los giros del robot, provoca movimientos erráticos del rotulador. Esto es más visible cuanto más agudo sea el ángulo de giro del robot.

Para solucionar este problema se piensa en colocar un nuevo punto de sujeción del rotulador en la plancha inferior del prototipo. Se diseña una pieza similar a la colocada en la parte superior con forma de tubo, pero bastante ajustada al diámetro del rotulador, para que no haya holgura y evitar así este movimiento.

La pieza que se diseñó y se colocó es la siguiente:

Figura 1 - 11 Pieza antivibración

1.8.2 Hardware

- Estudio de mercado

Se realizará una búsqueda en el mercado de los principales sensores y actuadores necesarios para el proyecto. Se realizará una comparación entre las diferentes alternativas que se encuentren y se seleccionará el dispositivo cuyas características y cuyo precio sea el más conveniente para el dispositivo final. Se deja fuera de este estudio aquellos dispositivos electrónicos que son encontrados con facilidad y a un bajo coste en el mercado local, tales como cables, pines o leds.

- Servomotores

Un servomotor es un accionador electrónico que permite controlar el ángulo de giro de manera directa. Por ello es amplia mente usado en electrónica. El rango de giro habitualmente va de los 0º a los 180º.

Consta internamente de un motor DC con una serie de engranajes reductores y un potenciómetro que permite determinar la posición del eje en todo momento. El conjunto de engranajes forma un mecanismo reductor que reduce la velocidad de operación del motor a costa de incrementar su torque.

Los servomotores realizan ellos mismo el control de posición del eje por lo que no es necesario mantener un control de manera externa [30].

Los servomotores, a través de una señal PWM, es decir, una señal eléctrica de ancho variable, se les puede controlar. Teniendo en cuenta que la señal de control ha de respetar el máximo y el mínimo del pulso PWM.

La frecuencia de transmisión de señales de pulso es de 50Hz, es decir, cada 20 ms el servo recibirá una señal que dependiendo su duración hará girar más o menos el motor.

Una señal con un pulso con un tiempo menor de 1.5 ms indicará un movimiento en sentido antihorario del servomotor, hasta una posición de 0°. Por el contrario, con un pulso mayor de 1.5 ms se producirá un movimiento en sentido horario hasta una posición de 90° [31].

Se utilizará un servomotor para realizar la acción de subir y bajar el rotulador de dibujo. Disponemos principalmente de tres modelos diferentes, cuyas características principales de cada uno de ellos son:

	Tensión de entrada (V)		Velocidad s/60°		Fuerza Kg.cm	
	Máx,	Mín,	Máx.	Mín.	Máx.	Mín.
SG90	6	4,8	0,1	0,09	1,8	1,7
MG90S	6	4,8	0,09	0,1	2,2	1.8
MG996R	6	4,8	0,15	0,17	11	9,4

Tabla 1 -1 Comparación características de servomotores (Fuente [33])

Dado su bajo coste y sus buenas características técnicas, que se acoplan perfectamente al dispositivo final, se elige el modelo SG90.

- Motores [33][34]
  - MOTORES GEARED DOWN MOTOR

Se trata de un tipo de motor de continua que pose una serie de engranajes internos con el fin de reducir la velocidad de movimiento del eje del motor, para así aumentar el par de fuerza que se entrega.

  - MOTORES BRUSHLESS

Son una variación también del motor de CC que no pose escobillas para rectificar la corriente. Utiliza circuitería electrónica para realizar la conmutación del campo magnético. Al no poseer escobillas alcanzan mayores velocidades que los motores CC convencionales.

- MOTORES PASO A PASO

Este tipo de motores cuentan con un estator constituido por un par de bobinas colocadas a 90 grados, y un rotor que consta de un imán unido al eje del motor.

Mediante un procesador externo se controlará el encendido y apagado de las bobinas para lograr el movimiento deseado en el motor.

- 28BYJ-48

Este es un tipo de motor paso a paso de un tamaño reducido. Tiene una tensión de entrada de 5V y el movimiento del motor se encuentra dividido en 64 pasos, con un ángulo de giro de cada paso de 5,62°. Además, para lograr un aumento del control del giro de estos motores, se les añade un circuito reductor 1/64, logrando reducir el giro del motor en cada paso hasta los 0,088°.

La velocidad de giro media de estos motores es de 7200 rpm con una fuerza de 34,3 mN.m.

En la siguiente tabla se hace un resumen de las principales características de los motores para hacer una comparación y valorar cual escoger, valorando cada una de ellas de 0 a 10, siendo 0 el valor más bajo y 10 el mayor:

	Características		**Control**	
	Velocidad	Fuerza/Par	Posición	Velocidad
Motor DC	8	3	1	1
Motor DC Geared Down	5	8	1	1
Motor brushless	9	3	1	1
Servo	3	9	10	10
Servo rotación continua	3	8	1	4
Motor paso a paso	5	5	10	10

Motor paso a paso BYJ48	1	4	10	10

Tabla 1- 2 Comparación de motores (Fuente [33] [35])

Dada su precisión y alto control de la posición y que no precisamos de alcanzar altas velocidades, se escoge para el dispositivo el motor 28BYJ-48.

- Drivers

Colocando una serie de drivers externos a los motores podremos mandar órdenes sobre estos, ya que los drivers permiten convertir la señal de baja corriente que se envía desde el Arduino en una señal de la corriente necesaria para alimentar a los motores.

Existen numerosos modelos de drivers en el mercado, con distintas características técnicas según la aplicación que se la vaya a dar. Para este proyecto se van a usar 2 drivers (uno por cada par prototipo) modelo ULN2003A. Este modelo es un circuito integrado constituido por 7 drivers idénticos. Cada uno de ellos cuenta con una serie de transistores colocados en puente Darlington para lograr un aumento de la ganancia de la señal.

Las características técnicas de este modelo son:

$V_{cemáx}$	$I_{máx}$	$V_{ce(sat)}$	V_{in}	I_{in}
50V	500mA	1-1.3V	1.7-2.2V	0.93-1.35mA

Tabla 1- 3 Características ULN2003A (Fuente [29])

1.8.3 Software

1.7.3.1 Programación Arduino Leonardo

La programación del microcontrolador ha sido realizada en el lenguaje de programación propio de Arduino, que está basado en C++ por lo que es fácil de utilizar.

El código está estructurado de tal forma que lo primero que nos encontramos son las constantes y datos que se van a emplear a lo largo del código. Dentro de esta zona cabe destacar las constantes que se van a utilizar para el domínio de los motores paso a paso. Estas variables están formadas por dos vectores con la secuencia de movimiento de estos. Como se ha dicho anteriormente (en el apartado de posición) se va a utilizar la secuencia de 2-Fases en la que se hace trabajar dos bobinas consecutivas en cada fase. Cómo se activan las bobinas para esta secuencia viene representado en la siguiente tabla:

Paso	A	B	A´	B´
1	ON	ON	OFF	OFF
2	OFF	ON	ON	OFF
3	OFF	OFF	ON	ON
4	ON	OFF	OFF	ON

Tabla 1- 4 Secuencia encendido/apagado motores (Fuente [28])

Los vectores fwd_secuencia y rev_secuencia representa esta tabla. Con la diferencia entre ellos que se encuentran invertidos ya que fwd_secuencia se utiliza para el movimiento hacia adelante y rev_secuencia se utiliza para el movimiento hacia atrás.

En las variables de las ruedas encontramos su diámetro y el número de pasos que es necesario que den los motores para que una rueda de una vuelta completa.

Con las variables del servomotor está incluido el pin que le corresponde del Arduino, la posición cuando el rotulador se encuentra levantado (PEN_UP) y la posición cuando el rotulador este tocando el papel (PEN_DOWN).

La siguiente zona de la estructura del código será la parte central constituida por el setup y el loop. En el setup se inicializan los pines de la placa de Arduino que se van a utilizar y se coloca el servomotor en su posición inicial de trabajo.

Loop es la parte del código que se está repitiendo constantemente. En esta zona se encuentra la llamada a todas las funciones que el robot es capaz de realizar.

Esta constantemente leyendo el puerto serial1 (comunicado vía inalámbrica con el equipo Xbee) esperando recibir una orden. Las ordenes que puede recibir son: adelante, retroceso, derecha, izquierda, triangulo, cuadrado, hexágono, circulo o polígono. Si recibe cualquier otra orden que no sea una de estas se mostrará un mensaje de valor introducido no correcto. La comprobación de que se ha introducido o no vía puerto serial se realiza con una estructura de if…else.

Si la orden que enviamos por el monitor serial es una orden correcta, de las citadas en el párrafo anterior, entraremos dentro de ese if, nos saldrá un mensaje en el monitor serial de lo que hemos escogido para dibujar y el código saltará a las funciones encargadas de ejecutar nuestra orden.

Esta sería la última parte de la estructura del código, un conjunto de funciones, a su vez divididas en dos clases, que son las encargadas de ejecutar el movimiento del robot. Las dos clases de funciones en las que se encuentra dividido: funciones de formas geométricas y funciones de ayuda.

Las funciones de formas geométricas son las encargadas de dibujar el cuadrado, triangulo, circulo, hexágono o polígono. Estas funciones nos van a pedir que introduzcamos, en centímetros, el valor del lado del polígono que queremos realizar. En la función polígono también se va a pedir que se introduzca el número de lados que queremos que tenga el polígono.

A continuación, en la función ángulo, a la que le llega como dato el número de lados, se calcula el valor de los ángulos del polígono con la siguiente fórmula:

$$ángulo = \frac{(Lados - 2) * 180}{Lados}$$

Es importante destacar que el ángulo que gira el robot no es el que salga de esta operación sino su suplementario que se calcula como:

$$ang.suplementario = 180 - ángulo$$

Estos valores, junto con la conversión de la distancia del lado en pasos del motor, que se realiza en una función de ayuda, vuelven a las funciones de formas geométricas para, mediante un bucle *for* ejecutar el movimiento. El bucle for se ejecutará tantas veces como lados tenga la figura a realizar.

La función círculo es algo diferente a las demás dada la complejidad de realizar un circulo con un robot que solamente se desplaza en línea recta. Se considera un polígono regular de 32 lados como una aproximación bastante certera a una circunferencia. Y se da por buena la aproximación de utilizar la apotema del polígono como radio de la circunferencia. Por tanto, la función pide al usuario que se introduzca el valor del radio de la circunferencia. Utilizando la función ángulo explicada anteriormente se calcula el valor del ángulo de un polígono de 32 lados. Con el valor de la apotema (por aproximación el valor del radio), y el valor del ángulo interno en radianes calculamos el valor de cada uno de los 32 lados que tiene que dibujar el robot. Para ello se utiliza la fórmula del cálculo de la apotema dejando como incógnita el valor del lado:

$$apotema = \frac{l}{2\tan(\frac{\alpha}{2})}$$

Donde α es el valor del ángulo y l la longitud del lado. Despejando esto último tendremos:

$$lado = apotema * 2 * \tan(\frac{\alpha}{2})$$

Por último, en la estructura de código nos encontramos las funciones de ayuda, que a su vez se encuentran divididas en dos clases: funciones de movimiento y funciones de cálculo.

Las funciones de movimiento son todas aquellas que intervienen para el movimiento del prototipo. Es decir, todas aquellas funciones encargadas de indicar una acción a realizar por el robot.

Las funciones de movimiento son:

- Retroceso (): función encargada del movimiento hacia atrás del prototipo. Recibe un parámetro que es la distancia que queremos que recorra. Esta distancia es envidada a una función de ayuda que la convierte en el número de pasos que los motores tienen que dar. La función de ayuda y como se convierte la distancia en número de pasos se explica más adelante.
- Adelante (): función encargada del movimiento hacia adelante del prototipo. Al igual que la función retroceso se recibe un parámetro con la distancia que queremos que recorra que convertimos en la función de ayuda en el número de pasos que los motores tienen que dar.
- Derecha (): función encargada de girar el prototipo hacia la derecha. Recibe un parámetro con el ángulo que se quiere girar. Usando la fórmula de la longitud del arco de circunferencia $l_{arco} = \frac{2*\pi*R*\alpha}{360}$ (donde R es el radio de las ruedas y α el ángulo de giro) se calcula la distancia que tienen que girar los motores. Este valor de la distancia se le pasa a la función de ayuda para que la convierta en el número de pasos que han de dar los motores.
  En el código se calcula de dos veces la longitud del arco. Primero en la variable rotación, se divide el ángulo entre 360 y después en la variable distancia se le multiplica por 2**π**.
- Izquierda (): función encargada de girar el prototipo hacia la izquierda. Realiza el mismo proceso que la función derecha. La única diferencia es el sentido que hace cada función que giren los motores.
- Arriba (): función encargada de subir el elemento de dibujo.
- Abajo (): función encargada de bajar el elemento de dibujo.

La función de cálculo es:

- Pasos (): función encargada de convertir la distancia en número de pasos que tienen que dar los motores. Para ello, primero dividimos la distancia a recorrer entre la longitud de circunferencia de la rueda (calculada como $l_{circunferencia} = \pi * \oslash_{\text{rueda}}$) obteniendo así el número de vueltas que tienen que dar.

Un motor necesita de 4076 pasos para dar una vuelta completa. Por tanto, multiplicando el número de vueltas por 4076 tendremos el número de pasos que tienen que dar los motores para recorrer la distancia que nosotros deseamos. Este valor es devuelto a las funciones de movimiento.

1.7.3.2 Librerías microcontrolador Arduino

Para facilitar el control y programación del servomotor se utiliza una librería. A continuación, se explica su funcionamiento y principales funciones:

1.7.3.2.1 Librería servomotor.

Esta biblioteca viene incluida con el IDE del Arduino para facilitar el dominio de los servomotores. Permite un control máximo de 12 servomotores. Para usar esta biblioteca debemos de añadir <*Servo.h*> y declarar el objeto Servo. Se selecciona el pin de salida como *Servo.attach (Nº de pin)* y la posición del servo como *Servo.write* (Posición), en que la posición va de 0 a 180 grados. [36]

Las funcionalidades de la librería son:

Attach (Pin): definición del pin a utilizar.

Attach (Pin, min, max): definición del pin a utilizar y de sus ángulos máximos de movimiento.

Write (ángulo): comando para indicar en que posición queremos que se coloque el servo.

writeMicroseconds (tiempo): definición del tiempo del pulso en microsegundos.

Read (): indica el ángulo de giro con el que se encuentra el servo.

1.7.3.2.2 Librería matemática

Esta librería permite el uso de funciones matemáticas básicas tales como senos, cosenos, tangentes… Se incluye al inicio del código como math.h. [37]

1.7.4 Comunicación

La comunicación inalámbrica del proyecto se va a realizar a través de los módulos de comunicación Xbee.

Xbee es el nombre comercial que reciben los módulos de comunicación fabricados por la compañía estadounidense Digi International, utilizando el protocolo de comunicación inalámbrico Zigbee. [38]

El protocolo de comunicación inalámbrico Zigbee está fundamentado en el estándar de comunicación inalámbrica IEEE 802.15.4 [39], cuyo objetivo principal son las comunicaciones que requieren bajas tasas de envíos de datos y maximizar el tiempo de vida de las baterías. [40]

Por tanto, es un protocolo especialmente útil para redes con sensores en entornos industriales, médicos y domóticos.

La frecuencia de operación de este protocolo se encuentra en los 868MHz para Europa (esta frecuencia forma parte de la banda ISM), 915 MHz en Norte América y 2.4GHz para todo el mundo. Las frecuencias mayores permiten una menor velocidad de envío y recepción de información, con unas velocidades medias de 20-40 kb/s.

Para cada frecuencia, a su vez, existen diferentes canales de frecuencia para el envío y recepción de datos. En el rango de Europa sólo soporta un canal entre 868 y los 868.6MHz; y diez canales en el rango de 902.0 y 928.0 MHz de los Estados Unidos. Ambos canales de frecuencia se consideran lo suficientemente cercanos como para usar el mismo hardware en ambos casos y así reducir costes de producción. Para la banda de frecuencia de 2.4GHz existen 16 canales comprendidos entre los 2.4 y los 2.4835 GHz con un espacio de 5 MHz entre canales lo que facilita el filtrado en la recepción y envío de datos. [41][42]

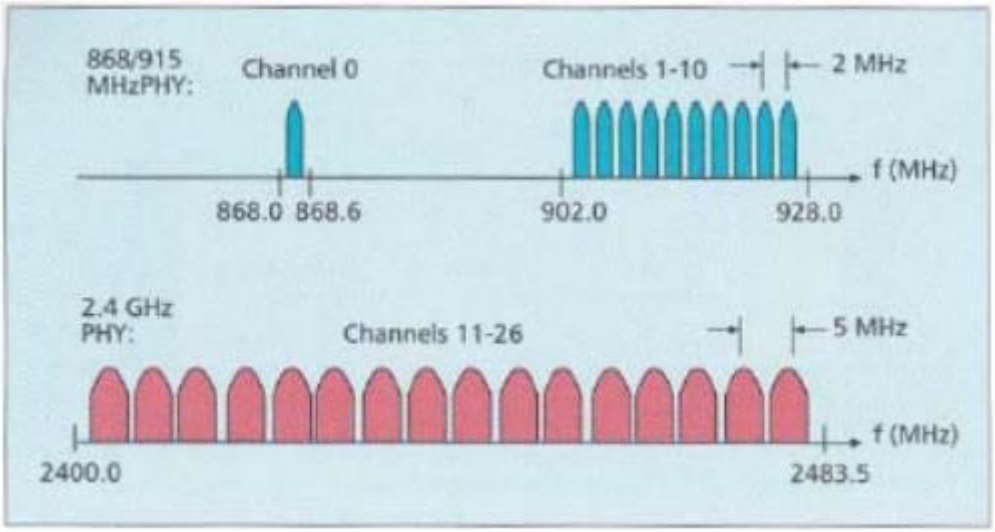


Figura 1 - 12 Banda de frecuencias (Fuente [42])

El alcance de la señal depende de la potencia del dispositivo y de la antena que se esté usando. Para bajas velocidades de transmisión de datos y dependiendo de la potencia del dispositivo (cuanta más potencia mayor rango) tenemos una zona de cobertura de entre 20-150m. Mientras que para velocidades de transmisión altas el rango de alcance disminuye en torno a 10-70m aproximadamente.

Las redes Zigbee pueden llegar a tener hasta 254 nodos, pero depende de cómo agrupemos estos nodos podemos crear hasta 255 conjuntos de nodos llegando a tener 64770 nodos. Para tener control sobre todos los nodos existen diferentes topologías de red:

- Estrella: existe un dispositivo central, con el que se hacen todas las comunicaciones, al que se encuentran conectados todos los dispositivos. Esta topología es considerada de baja potencia al no tener que estar todos los dispositivos comunicando al mismo tiempo.
- Malla: Cada dispositivo se encuentra conectado a todos los dispositivos. No existe un dispositivo central. En esta topología un mensaje puede recorrer diferentes caminos para llegar a un punto.
- Árbol: se trata de una mezcla de las dos topologías anteriores.

Dependiendo del diseño final de nuestra aplicación, se escogerá una topología u otra. Por ejemplo, si se requiere de conexiones de baja potencia se escogerá tipo estrella, por el contrario, si necesitamos crear un perímetro con mayor cobertura y interconectado entre si escogeremos una topología tipo malla.

En el protocolo de comunicación ZigBee existen tres tipos distintos de dispositivos: el coordinador, el router y los dispositivos finales (end device).

- Coordinador: dispositivo encargado de crear la red y de establecer el canal de comunicación. Crea un número de identificación único para toda la red. Una vez establecida la red puede incorporarse a la arquitectura de esta como un router más.
- Router: Determina la mejor ruta para enrutar un paquete de información entre dos nodos.
- End Device: Dispositivo final de la red que solo puede recibir paquetes de información. Es decir, no tiene capacidad de enrutar paquetes para otros nodos.

En el tema de seguridad, el protocolo ZigBee utiliza la encriptación AES de 128 bits que facilita la encriptación y autentificación de las comunicaciones. Además, se utiliza un mecanismo de red denominado Trust Center que a través de una clave de enlace y una clave de red proporciona un mecanismo de seguridad.

Módulos Xbee

Digi es una compañía especializada en la supervisión y gestión remota de las comunicaciones de las que depende el IoT (Internet de las cosas), la cual ha desarrollado los módulos de comunicación por radio frecuencia XBEE. Existen diferentes modelos de estos módulos de comunicación que están catalogados por series:

XBEE Series 1 ó XBee 802.15.4: Son los módulos más sencillos de operar ya que no precisan de ser configurados previamente. Son los más recomendados para iniciarse con ellos y para realizar comunicaciones Punto-a-Punto. El Hardware de esta serie con los de otras series más avanzadas no son compatibles.

XBee Znet 2.5 ó Series 2: Actualmente Retirado: Estos módulos sí necesitan de una preconfiguración antes de ser usados. Dependiendo del firmware configurado en los módulos pueden trabajar en modo transparente o en modo API. Estos módulos no se encuentran en el mercado actualmente.

XBee ZB: Modelo que viene a reemplazar al Znet 2.5 con un nuevo firmware que le permite ser más compatible con otros dispositivos. Son llamados de manera popular módulos de Serie 2.

Xbee 2B (módulos más actuales de Serie 2): Poseen mejoras en el firmware para mejorar su rendimiento energético. Son compatibles con el firmware del modelo anterior pero no con el del Znet 2.5.

Por lo tanto, si queremos configurar una red de comunicación entre diferentes dispositivos lo más recomendable es utilizar modelos de la Serie 2, teniendo en cuenta las compatibilidades de firmware que hay entre ellos. Por otro lado, si lo que buscamos es una comunicación entre dos puntos (Punto-a-Punto) lo más recomendable, por ser los más sencillos de configurar y usar, es emplear el módulo de la Serie 1.

XCTU

XCTU (XBee Configuration and Test Utility): se trata de un software gratuito y multiplataforma (compatible con Windows, MacOS y Linux) que a través de una interfaz gráfica nos permite configurar, inicializar, actualizar y probar el funcionamiento de los módulos XBee. [16]

Este software se comunica mediante puerto serie con los módulos, permitiendo ver de forma rápida y sencilla todos los valores del módulo y una definición de cada uno.

El software XCTU se descarga directamente de la página de la compañía Digi, junto con los drivers necesarios para la lectura de los puertos USB.

Al abrir el programa nos encontramos con la siguiente interfaz:

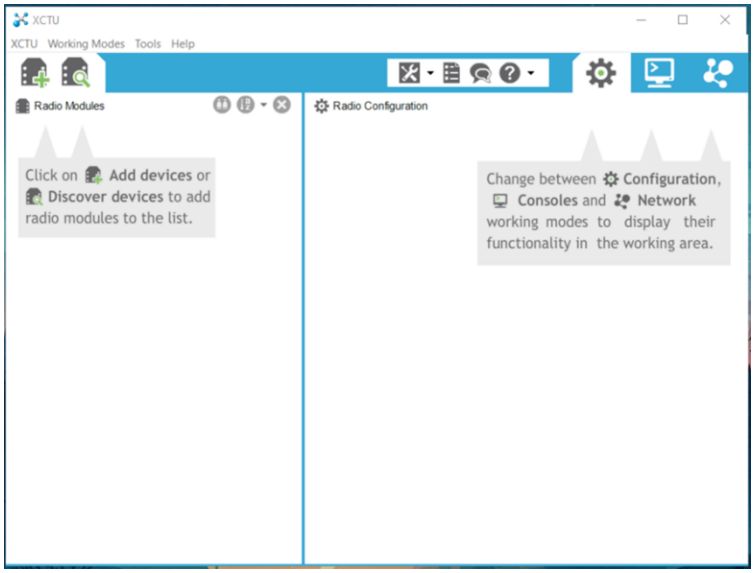


Figura 1 - 13 Captura de pantalla de inicio de Xbee

Haciendo click sobre el dibujo con una lupa de la parte superior izquierda, se realiza la búsqueda de módulos que añadir a la lista de dispositivos.

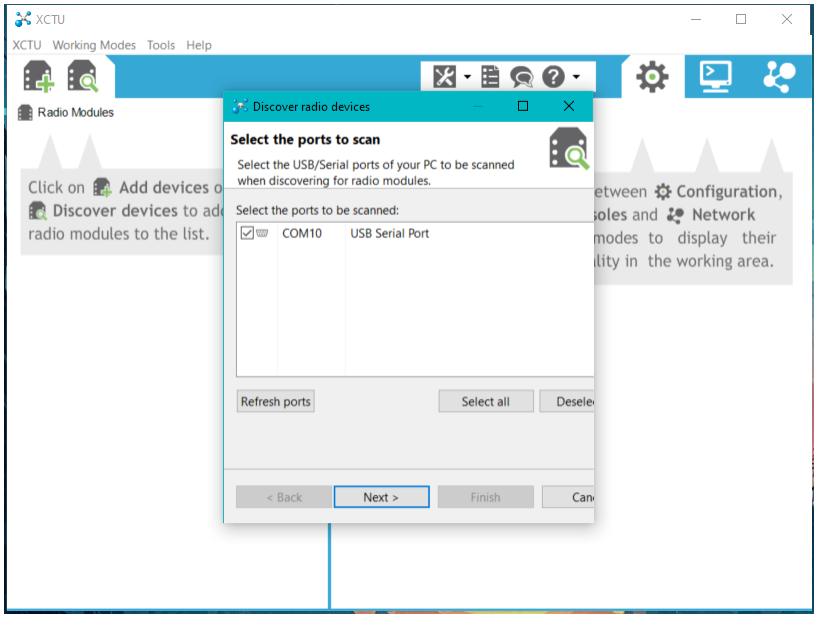


Figura 1 - 14 Captura de pantalla selección de puerto

Se abrirá una nueva pestaña en la que tenemos que escoger el puerto en el que está conectado el módulo.

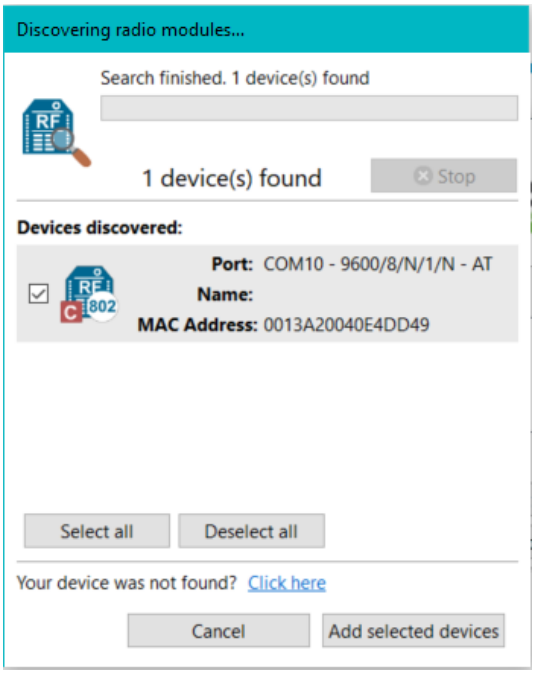


Figura 1 - 15 Captura de pantalla selección de dispositivo

Al seleccionar el puerto y los parámetros correctos en la configuración del puerto USB, se realizará la búsqueda de los módulos y seleccionaremos el que nos interese.

Una vez hayamos encontrado y seleccionado el módulo correcto (el cual aparecerá en la columna de la izquierda de la interfaz) haciendo doble click sobre él se nos abrirán sus características de configuración (columna de la derecha).

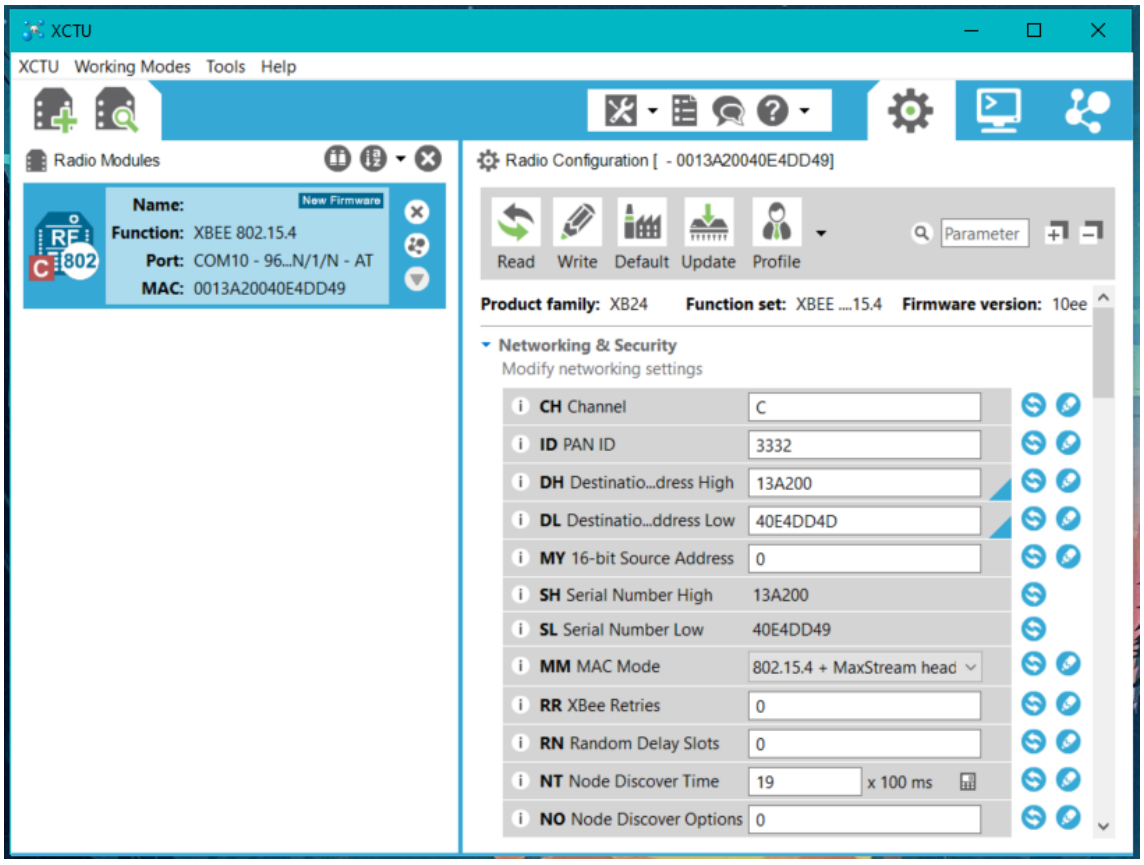


Figura 1 - 16 Captura de pantalla características del módulo

Cómo se comunican los módulos Xbee

Cuando se envía información entre dos módulos XBee podemos ver dos formas distintas de comunicación:

- Comunicación inalámbrica: comunicación que se establece entre los distintos módulos que forman una misma red y usan la misma frecuencia.
- Comunicación serie: comunicación que se establece entre el microcontrolador o Pc y el módulo XBee a través de un puerto serie.

Figura 1 - 17 Comunicación Xbee-ordenador (Fuente [40])

Comunicación inalámbrica

Para establecer la comunicación inalámbrica entre dos o más módulos han de formar parte de una misma red. Los dos parámetros más importantes para definir la red son:

- Channel (CH): frecuencia para comunicar un mismo canal dentro de la red.
- Personal Area Network Identifier (ID): número de identificación de la red.

Por tanto, a la hora de configurar un módulo pondremos los mismos canales (CH) y la misma red (ID) en todos aquellos módulos que queramos que se comuniquen entre ellos.

A su vez, cada módulo tiene una dirección de identificación propio dentro de la red, denominada dirección MAC. La dirección MAC está compuesta por 64 bits divididos en dos parámetros. El Serial Number High (SH) y el Serial Number Low (SL). El valor del SH es el mismo para todos los dispositivos ya que identifica a la compañía Digi (0013A200) y el SL viene escrito en la parte posterior de cada dispositivo. La dirección 000000000000FFFF está reservada para mensajes de broadcast.

En resumen, para que un módulo A se comunique con un módulo B hay que poner dentro del software XCTU la siguiente información:

- Definir el módulo A, por ejemplo, como coordinador y el módulo B, por ejemplo, como receptor (end device).

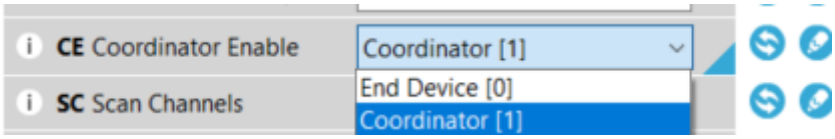


Figura 1 - 18 Definición de coordinador y receptor

- Establecer en ambos módulos la misma red (ID) y el mismo canal (CH)

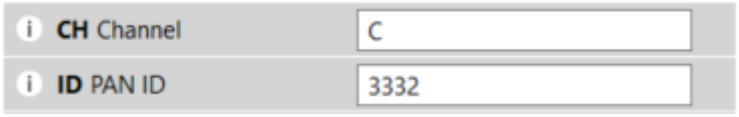


Figura 1 - 19 Definición de la red y el canal

- En el apartado Destination Address High escribir el SH de ambos módulos, al ser los dos de Digi será el mismo para los dos: 0013A200
- En el apartado Destination Address Low escribir el SL del módulo B, si estamos configurando el módulo A y viceversa. Este valor viene escrito en la parte trasera de los módulos.

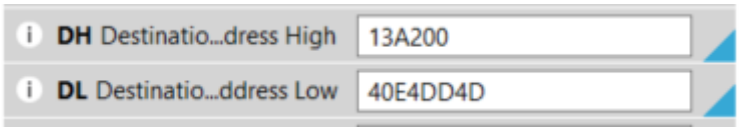


Figura 1 - 20 Configuración de las direcciones

Con esto ya quedaría establecida la comunicación entre dos módulos.

Comunicación Serie

La comunicación serie se establece cuando un módulo no opera de manera independiente, sino que está recibiendo información a través del puerto serie procedente de un microcontrolador o PC. De esta manera, se establecen dos modos de trabajo:

- **Modo transparente (Aplicación transparente):** El módulo envía la información tal cual la recibe en el puerto serie. Es la mejor forma de empezar a trabajar con los módulos.
  Este modo de operación tiene muchas limitaciones si estamos trabajando en una red con varios dispositivos, ya que sería necesario configurar la dirección de destino antes de enviar cada mensaje. Por otro lado, es muy idóneo para comunicaciones entre dos puntos dada su sencillez.

- **Modo API (Aplicación de programación):** El envío y recepción de datos se produce siguiendo un protocolo definido por el usuario previamente. De esta forma, podemos crear una red de comunicaciones más compleja.

  Este modo es recomendado para redes grandes en las que un microcontrolador externo se encarga de crear la trama a enviar entre los distintos dispositivos y así no perder tiempo entrando manualmente en cada uno de ellos para realizar su configuración.

XBee Shield

Para poder acoplar un módulo XBee a una placa de Arduino es necesario utilizar un aparato llamado XBee Shield. Estas placas tienen dos modos de funcionamiento; modo XBee o modo USB. Estos se escogen con un jumper o un switch situado en la placa. [44]

Si el jumper está colocado en la posición XBee, el pin DOUT (pin de salida) perteneciente del Xbee, hará conexión con el pin RX del Arduino, y el pin DIN con el pin TX también del Arduino. De esta forma los pines RX y TX aún siguen conectados a los pines TX y RX del dispositivo XBee haciendo que el microcontrolador pueda transmitir datos vía USB al ordenador, pero no pueda recibir información desde el ordenador por el USB, solo podrá recibir datos a través del módulo XBee.

Con el jumper colocado en modo USB el pin DOUT del módulo XBee hará conexión con el pin RX del XBee (no del microcontrolador), y el DIN del XBee hará conexión con el pin TX del XBee. Permitiendo así comunicar directamente con el ordenador el dispositivo Xbee. Pero para que así suceda es necesario extraer o anular el microprocesador de la placa Arduino. Si esto no se hace se podrá comunicar con el ordenador, pero no podremos comunicarnos con el Xbee.

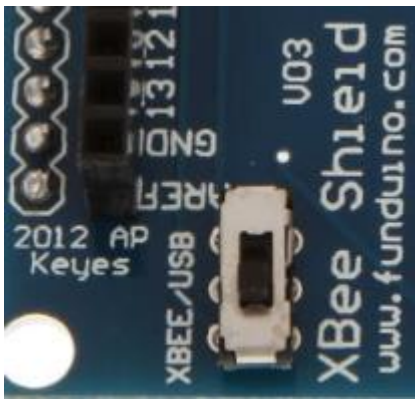


Figura 1 - 21 Interruptor Xbee Shield

1.7.5 Montaje

1.7.5.1 Conexión de pines

En la siguiente tabla se representa como se distribuyen las conexiones del Arduino. Cada uno de los motores ocupa 4 pines digitales. Mientras que el servomotor solo precisa de un pin.

Conexiones de pines	
Pines digitales	
D2	Motor derecho
D3	
D4	
D5	
D8	Motor Izquierdo
D9	
D10	
D11	
D13	Servomotor

Tabla 1- 5 Conexiones de pines

1.9 Resultados finales

El resultado final de este proyecto es la creación de un dispositivo con comunicación inalámbrica que se puede desplazar en cualquier dirección mediante una serie de órdenes básicas.

El prototipo final, como se analizó en apartados anteriores, consta de 2 ruedas motrices colocadas en la parte delantera y un punto de apoyo a modo de rueda libre, constituido por una bola de acero. Este diseño permite desplazarse libremente en cualquier dirección y de forma precisa al robot sin sufrir problemas de que las ruedas patinen o no tengan la suficiente fuerza como para realizar el movimiento.

Para realizar la sujeción para el rotulador se diseñó una pieza capaz de mantenerlo en posición vertical y mediante un servomotor se realiza el movimiento de subir o bajar el rotulador para dibujar. Además, en la plancha inferior del prototipo se colocó una pieza con forma de tubo, para evitar movimientos y vibraciones en el rotulador a la hora de dibujar cuando el prototipo está en movimiento.

La comunicación se realiza mediante dos módulos Xbee, un receptor y un coordinador. El receptor está colocado en el prototipo y el coordinador está conectado a un ordenador con el programa de control de los Xbee llamado XCTU con el que se envían las ordenes al receptor.

Las ordenes que el prototipo es capaz de interpretar son las siguientes:

- Adelante / Retroceso: el programa pedirá al usuario que introduzca que distancia (en cm) quiere que se desplace. Una vez introducida el robot se desplaza hacia adelante o hacía atrás, según lo seleccionado.
- Derecha / Izquierda: el programa pedirá al usuario que introduzca que ángulo (en grados) quiere que gire. Una vez introducido el robot gira hacia la izquierda o derecha, según lo seleccionado.
- Arriba / Abajo: el robot sube o baja el rotulador para dibujar.
- Triangulo: el programa pedirá que introduzcas la longitud del lado en cm. Una vez introducido el robot realizará un triángulo equilátero con el valor del lado el que se haya introducido.
- Cuadrado: el programa pedirá que introduzcas la longitud del lado en cm. Una vez introducido el robot realizará un cuadrado.
- Hexágono: el programa pedirá que introduzcas la longitud del lado en cm. Una vez introducida el robot realizará un hexágono.
- Circulo: el programa pedirá que introduzcas el radio de la circunferencia en cm. Una vez introducido el robot realizará una circunferencia. En este caso, tal y como se ha explicado en

un apartado anterior, la circunferencia se aproxima a un polígono de 32 lados con una apotema igual al radio de la circunferencia que se quiere dibujar.

- Polígono: el programa primero pedirá que se introduzca el número de lados que se quiere que tenga el polígono, y segundo pedirá que se introduzca la distancia en cm que se quiere que tenga cada lado. Una vez introducidos todos los datos, el programa dibujará el polígono.
- Para cualquier palabra o valor que se introduzca y que el programa no sea capaz de interpretar saldrá un mensaje de error y se pedirá al usuario que vuelva a introducir el dato.

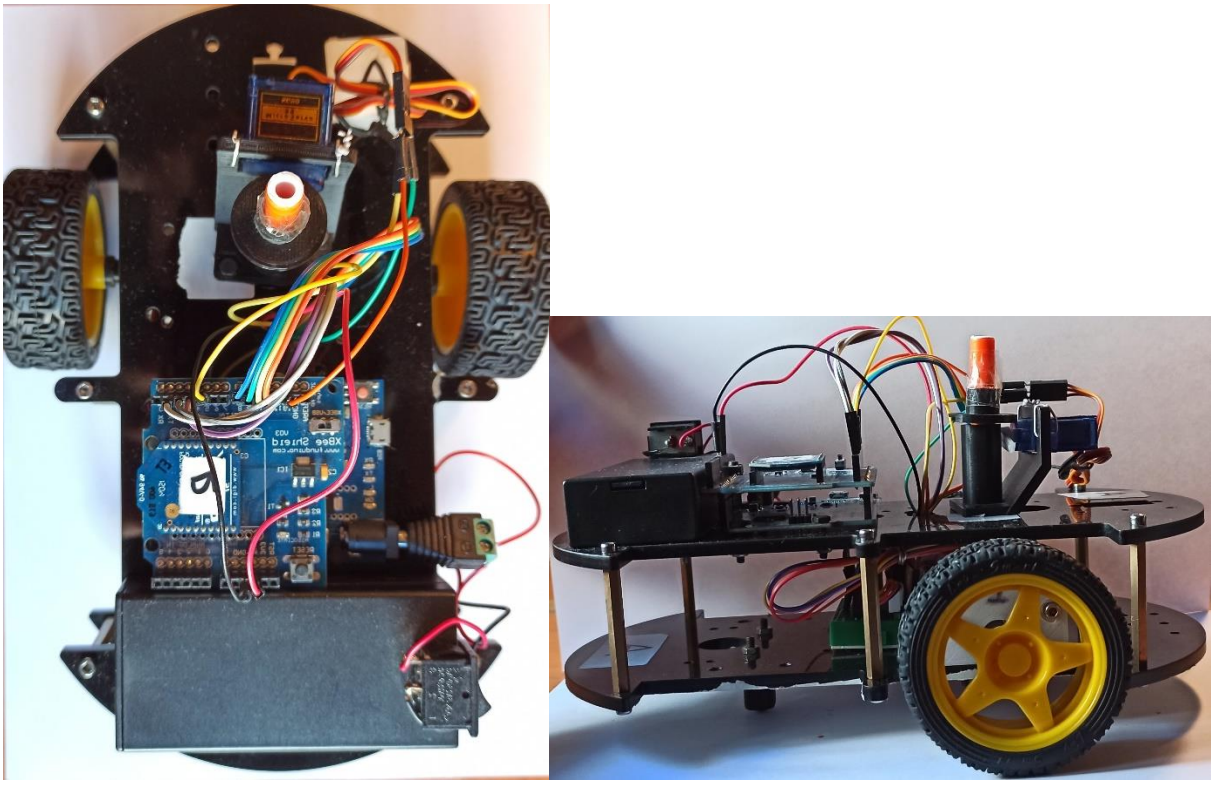


Figura 1 - 22 Imágenes del robot finalizado

2 Anexo

2.1 Anexo I: Hojas de características de los actuadores.

2.1.1 Servomotor SG90

Servomotor SG90 [32] [45]				
Características eléctricas				
	Mínimo	Medio	Máximo	Unidades
Voltaje de alimentacion	4,8	5	6	V
Caracteristicas mecánicas				
Par	1,8			Kg.cm
Velocidad	0,1			s/60
Largo	22,2			mm
Alto	11,8			mm
Ancho	31			mm
Otros				
Banda muerta	10			us
Temperatura ambiente de operación		0	55	ºC
Comunicación	PWM			

Tabla 2 - 1 Hoja de datos Servomotor SG90 (Fuente [44] [31])

Figura 1 - 23 Servomotor SG90 (Fuente [31])

2.1.2 Motores 28BYJ-48

28BYJ-48 [46]		
		Unidades
Tensión nominal	5	VDC
Número de fases	4	
Relación de variación de velocidad	1/64	
Ángulo de paso	5.625/64	
Frecuencia	100	Hz
Resistencia DC	50	Ω
Frecuencia de tracción inactiva	>600	Hz
Frecuencia de traccion en relentí	>1000	Hz
Torque	>34,3	mN.m(120Hz)
Par de posición	>34,3	mN.m(120Hz)
Fricción de torque	600-1200	gf.cm
Fuerza de torque	300	gf.cm
Ruido	<35	dB (120Hz)

Tabla 2 - 2 Hoja de datos motor 28BYJ-48 (Fuente [27] [45])

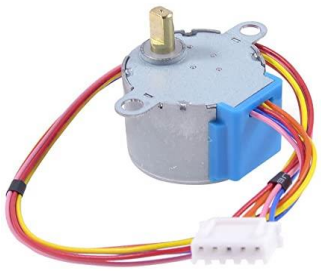


Figura 1 - 24 Motor paso a paso 28BYJ-48 (Fuente [28])

2.1.3 ULN 2003A

ULN 2003A [30]				
Simbolo	Parámetro	Min	Max	Unidad
Vce(sat)	Tensión colector-emisor saturación		1,3	V
Ii(on)	Corriente entrada		1,35	mA
Ii(off)	Corriente entrada	50		µA
Vi(on)	Tension entrada		3	V
hfe			25	pF
Ic	corriente colector		500	mA
Ib	corriente base		25	mA
Ci	Capacidad entrada		25	pF

Tabla 2 - 3 Hoja de datos ULN2003A (Fuente [29])

Figura 1 - 25 ULN2003A (Fuente [29])

2.2 Anexo II: Hoja de datos microcontrolador

2.2.1 Arduino Leonardo

Arduino Leonardo [47]		
		Unidades
Microcontrolador	ATmega32u4	
Tensión de operación	5	V
Voltaje de entrada (recomendado)	7-12	V
Voltaje de entrada (limites)	6-20	V
Entradas digitales	20	
Canales PWM	7	
Entradas analógicas	12	
Corriente por I/O pin	40	mA
Corriente por 3,3V pin	50	mA
Memoria Flash	32	KB
SRAM	2,5	KB
EEPROM	1	KB
Velocidad reloj	16	MHz
Largo	68,6	mm
Ancho	53,3	mm
Peso	20	g

Tabla 2 - 4 Hoja de datos Arduino Leonardo (Fuente [46])

Figura 1 - 26 Arduino Leonardo (Fuente [47])

2.3 Anexo III: Hoja de datos módulos de comunicación

2.3.1 Xbee Serie 1

Xbee serie 1 [49]		
		Unidades
Rendimiento		
Velocidad datos	250	kbps
Rango en interior	30	m
Rango en exterior	100	m
Potencia transmisison	1	mW
Sensibilidad recepción	-92	dBm
Digi Hadware	s1	
Caracteristicas		
Banda frecuencia	2,4	GHz
Velocidad datos serie	1200-250	bps-kbps
ADC inputs	10-bit	
Digital I/O	8	
Seguridad		
Encriptación	128-bit AES	
Requirimientos de potencia		
Voltage alimentación	2,8-3,4	VDC
Corriente transmison	45	mA-3,3VDC
Corriente recepción	50	mA-3,3VDC

Tabla 2 - 5 Hoja de datos Xbee serie 1 (Fuente [48])

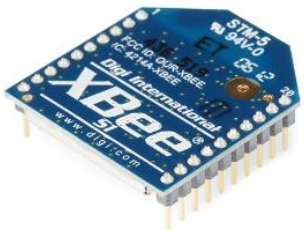


Figura 1 - 27 Módulo Xbee (Fuente [48])

2.3.2 Xbee Shield

Xbee Shield [50]				
Especificaciones				
Tamaño PCB	54,9 x 58,8 x 1,6 mm			
Indicaciones	PWR State, DI, DO			
Potencia alimentacion	5V DC			
Protocolo comunicación	UART/Xbee			
Características Electricas				
	Min	Type	Max	Unit
Tensión alimentación	4,5	5	5,5	VDC
Tension entrada VH	4,5	5	5,5	V
Tension entrada VL	-0,3	0	0,5	V
Corriente	-	20	40	mA

Tabla 2 - 6 Hoja de datos Xbee Shield (Fuente [49])

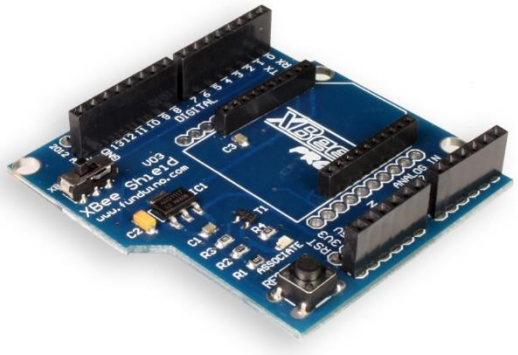


Figura 1 - 28 Xbee Shield (Fuente [49])

2.4 Anexo IV: Flujograma

Flujograma de representación de la secuencia del código

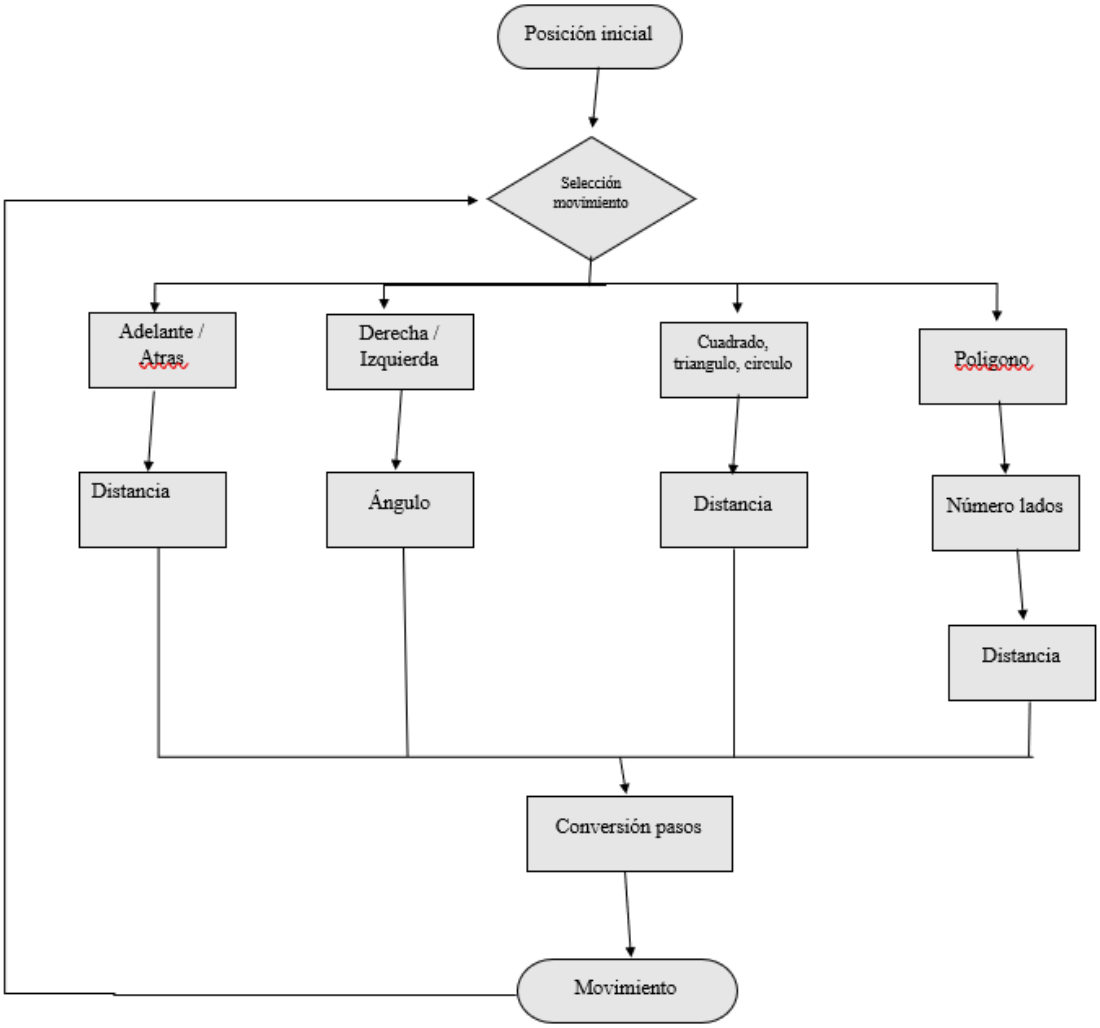


Figura 1 - 29 Flujograma del código

2.5 Anexo V: Código Fuente

A continuación, se incluye el código que se ha usado para programar el prototipo:

```
1. #include <Servo.h>
2.  #include<math.h>
3.
4. #define pi 3.141592
5. //Definición de pines
6.
7. // setup motor
8.       //Orden de la secuencia naranja->rosa->azul->amarillo
9. int motor_1_pin[] = {11, 9, 8, 10};
10.     int motor_2_pin[] = {2, 4, 5, 3};
11.
12.     int fwd_secuencia[][4] =  {{1, 0, 1, 0},  //Secuencia de
   encendido bobinas del motor para movimiento adelante,secuencia de 2-
   Fases
13.                               {0, 1, 1, 0},
14.                               {0, 1, 0, 1},
15.                               {1, 0, 0, 1}};
16.
17.     int rev_secuencia[][4] =  {{1, 0, 0, 1},  //Secuencia de
   encendido bobinas del motor para movimiento hacia atrás, secuencia
   de 2-Fases
18.                               {0, 1, 0, 1},
19.                               {0, 1, 1, 0},
20.                               {1, 0, 1, 0}};
21.     float diametro_rueda=5.2; //diametro rueda en cm
22.     int stepsPerRev = 4076;  // pasos para una vuelta completa
23.     int delay_time = 2;      // tiempo entre pasos en ms
24.
25.     //setup Servo
26.     Servo myservo;  // crea el objeto servo
27.     int pinSERVO= 13; //Vinculamos el servo al pin 13
28.     int pos_servo = 0;    // posicion del servo
29.     int PEN_UP= 180;  //Ángulo del servo cuando esta arriba
30.     int PEN_DOWN = 110; //Ángulo del servo cuando esta abajo
31.
32.     int i=0;            //Variables de ayuda para el código
33.     int lados=0;        //Variable para indicar el numero de lados
   de las figuras geométricas
34.     float avance=0;     //variable para indicar la distancian que
   queremos desplazar
35.     float giro=0;     //variable para indicar el angulo a girar
36.
37.     void setup() {
38.       randomSeed(analogRead(1));
39.
40.       Serial.begin(9600); //Arduino
41.       Serial1.begin(9600); //XCTU
42.       Serial1.setTimeout(2000);
43.       for(int pin=0; pin<4; pin++){
44.         pinMode(motor_1_pin[pin], OUTPUT); //Declaramos como
   salidas los puertos de los motores
45.         digitalWrite(motor_1_pin[pin], LOW);
46.         pinMode(motor_2_pin[pin], OUTPUT);
47.         digitalWrite(motor_2_pin[pin], LOW);
48.       }
49.       myservo.attach(pinSERVO);
```

```
50.
51.     Serial1.println("Inicio del programa");
52.     arriba();
53.
54.     }
55.     void loop(){
56.       if (Serial1.available()){
57.
58.         String data = Serial1.readStringUntil('\n');
59.        /*Menu de selección de lo que queremos hacer. Introducioendo
   las palabras:  adelante, retroceso o atras, derecha e izquierda
   indicaremos una direccion de movimiento.
60.         Tambien podemos seleccionar formas geometricas ya
   establecidas como triangulo, cuadrado, hexágono o circulo*/
61.
62.         if (data == "adelante"){
63.          Serial1.println("Nos desplazamos hacia adelante");
64.          avance= distancia();
65.           abajo();
66.           adelante(avance);
67.
68.         }
69.         else if (data == "retroceso"){
70.           Serial1.println("Nos desplazamos hacia atrás");
71.           avance= distancia();
72.           abajo();
73.           retroceso(avance);
74.
75.         }
76.         else if (data == "derecha"){
77.            Serial1.println("Nos desplazamos hacia la derecha");
78.            giro= angulo();
79.            abajo();
80.           derecha(giro);
81.
82.         }
83.         else if (data == "izquierda"){
84.            Serial1.println("Nos desplazamos hacia la izquierda");
85.            giro =angulo();
86.            abajo();
87.           izquierda(giro);
88.
89.         }
90.          else if (data == "arriba"){
91.
92.           arriba();
93.         }
94.          else if (data == "abajo"){
95.
96.           abajo();
97.         }
98.         else if (data == "triangulo"){
99.            Serial1.println("Quieres dibujar un triangulo
   equilatero");
100.           abajo();
101.          triangulo();
```

```
102.
103.
104.         }
105.         else if (data == "cuadrado"){
106.            Serial1.println("Quieres dibujar un cuadrado");
107.            abajo();
108.           cuadrado();
109.
110.         }
111.          else if (data == "hexagono"){
112.            Serial1.println("Quieres dibujar un hexágono");
113.            abajo();
114.           hexagono();
115.
116.         }
117.          else if (data == "circulo"){
118.            Serial1.println("Quieres dibujar un circulo");
119.            abajo();
120.           circulo();
121.
122.         }
123.          else if (data == "poligono"){
124.          Serial1.println();
125.         Serial1.println("De cuantos lados quieres que tenga ");
126.        delay(2000);
127.         float l = (char)Serial1.readString().toFloat();
128.          Serial1.println();
129.         Serial1.println("El lado de cuantos cm lo quieres");
130.        delay(2000);
131.         float cm = (char)Serial1.readString().toFloat();
132.         abajo();
133.         poligono(l, cm);
134.
135.         }
136.         else {
137.         Serial1.println("Valor introducido no correcto F ");
138.         }
139.
140.       }
141.     }
142.
143.
144.     //--------------Funciones formas geométricas-------------------
    --
145.
146.
147.     int cuadrado (){                                  //Funcion para
    hacer un cuadrado
148.       Serial1.println("Dibujamos un cuadrado");
149.       lados =4;
150.       avance= distancia();
151.      float ang= angulo(lados);
152.       for (i=0; i<=3; i++){
153.        delay(100);
154.       izquierda(ang);
155.       delay(100);
```

```
156.      adelante(avance);
157.
158.      }
159.    }
160.    int triangulo(){                                  //Funcion para
  hacer un triangulo
161.      Serial1.println("Dibujamos un triangulo");
162.      lados =3;
163.      avance= distancia();
164.      float ang= angulo(lados);
165.      for (i=0; i<=2; i++){
166.      delay(100);
167.      izquierda(180-ang);  //valor del angulo suplementario que
  queremos girar
168.      delay(100);
169.      adelante(avance);
170.      }
171.    }
172.    int hexagono(){                                    //Funcion
  para hacer un hexagono
173.        Serial1.println("Dibujamos un hexagono");
174.        lados=6;
175.        avance= distancia();
176.        float ang= angulo(lados);
177.        for (i=0; i<=5; i++){
178.        delay(100);
179.      izquierda(180-ang);
180.      delay(100);
181.      adelante(avance);
182.        }
183.    }
184.    int circulo(){                                   //Funcion para
  hacer un circulo
185.      Serial1.println("Dibujamos un circulo");
186.      lados= 32;
187.      avance= distancia();                     // Aproximamos el
  radio de la circunferecnia por la apotema de un poligono de 32 lados
188.      float giro = angulo(lados);
189.      float ang = (2*pi)/lados;                //Calculamos el
  angulo central en radianes
190.      float lado = avance*2*tan((ang/2));      // Con el valor del
  angulo central y la apotema (radio por aproximación) calculamo el
  valor de cada lado (por apróx. arcos de circunferencia)
191.      for (i=0; i<=32; i++){
192.        delay(100);
193.        adelante(lado);
194.        delay(100);
195.        izquierda(180-giro);
196.      }
197.    }
198.
199.    int poligono (float lad, float cm){   //funcion para hacer un
  poligono regular de n lados
200.      float ang= angulo(lad);
201.      for (i=0; i<=(lad-1); i++){
202.        delay(100);
```

```
203.         izquierda(180-ang);
204.         delay(100);
205.         adelante(cm);
206.       }
207.     }
208.
209.
210.     // ---------Funciones de Ayuda----------
211.
212.
213.     //Funciones de cálculo
214.
215.     float distancia(){                                          //Pedim
   os el valor de la distancia que se quiere avanzar
216.         Serial1.println();
217.         Serial1.println("Que distancia quieres que se mueva");
218.        delay(2000);
219.         float dis = (char)Serial1.readString().toFloat();
220.       return dis;
221.     }
222.
223.     float angulo(){                                              //Ped
   imos el valor del angulo que se quiere girar
224.         Serial1.println();
225.         Serial1.println("Que angulo quieres girar");
226.        delay(2000);
227.         float angulo = (char)Serial1.readString().toFloat();
228.
229.       return angulo;
230.     }
231.
232.     float angulo (float lados){             //Con esta función
   calculamos el angulo de giro para las formas geometricas
233.       float angulo = ((lados-2)*180)/lados;
234.
235.       return angulo;
236.
237.     }
238.
239.     float pasos (float distancia){ //Función encargada de
   transformar la distancia en pasos de motor
240.       float vueltas = (distancia/10)/(3.1415*diametro_rueda);  //Ca
   lculamos el numero de vueltas
241.       float pasos = 4076*vueltas;   //Convertimos las vueltas en
   pasos
242.       return pasos;
243.     }
244.
245.     //Funciones de movimiento
246.
247.     void retroceso(float distancia){ //Función de avance hacia
   adelante
248.       float paso = pasos(distancia);
249.       for (float i=0; i<paso; i++){
250.         for (int j=0; j<4; j++){
251.           for (int pin=0; pin<4; pin++){
```

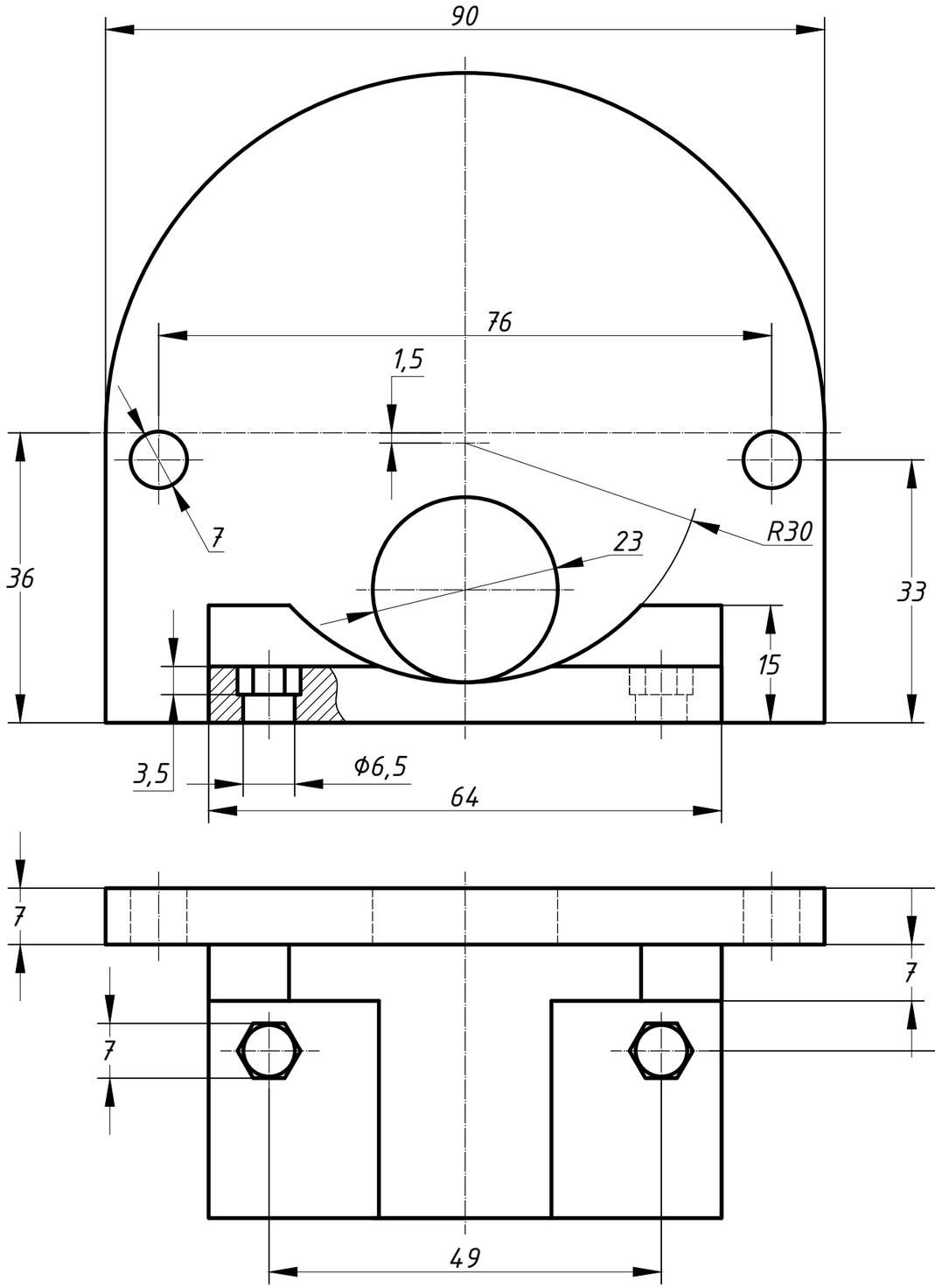
```
252.            digitalWrite(motor_1_pin[pin], rev_secuencia[j][pin]);
253.            digitalWrite(motor_2_pin[pin], fwd_secuencia[j][pin]);
254.          }
255.          delay(delay_time);
256.        }
257.      }
258.    }
259.
260.    void adelante(float distancia){
261.       float paso = pasos(distancia);
262.      for (int i=0; i<paso; i++){
263.        for (int j=0; j<4; j++){
264.          for (int pin=0; pin<4; pin++){
265.            digitalWrite(motor_1_pin[pin], fwd_secuencia[j][pin]);
266.            digitalWrite(motor_2_pin[pin], rev_secuencia[j][pin]);
267.          }
268.          delay(delay_time);
269.        }
270.      }
271.    }
272.
273.    void izquierda(float grados){
274.
275.      float rotacion = (grados*2)/360;
276.      float distancia = rotacion*3.1415*2*3.45;
277.      float paso = pasos(distancia);
278.      for (int i=0; i<paso; i++){
279.        for (int j=0; j<4; j++){
280.          for (int pin=0; pin<4; pin++){
281.            digitalWrite(motor_1_pin[pin], rev_secuencia[j][pin]);
282.            digitalWrite(motor_2_pin[pin], rev_secuencia[j][pin]);
283.          }
284.          delay(delay_time);
285.        }
286.      }
287.    }
288.
289.    void derecha (float grados){
290.
291.      float rotacion = (grados*2)/360;
292.      float distancia = rotacion*3.1415*2*3.45;
293.      float paso = pasos(distancia);
294.      for (int i=0; i<paso; i++){
295.        for (int j=0; j<4; j++){
296.          for (int pin=0; pin<4; pin++){
297.            digitalWrite(motor_1_pin[pin], fwd_secuencia[j][pin]);
298.            digitalWrite(motor_2_pin[pin], fwd_secuencia[j][pin]);
299.          }
300.          delay(delay_time);
301.        }
302.      }
303.    }
304.
305.    void arriba(){
306.      Serial1.println("Lapiz Arriba");
307.      myservo.write(PEN_UP);
```

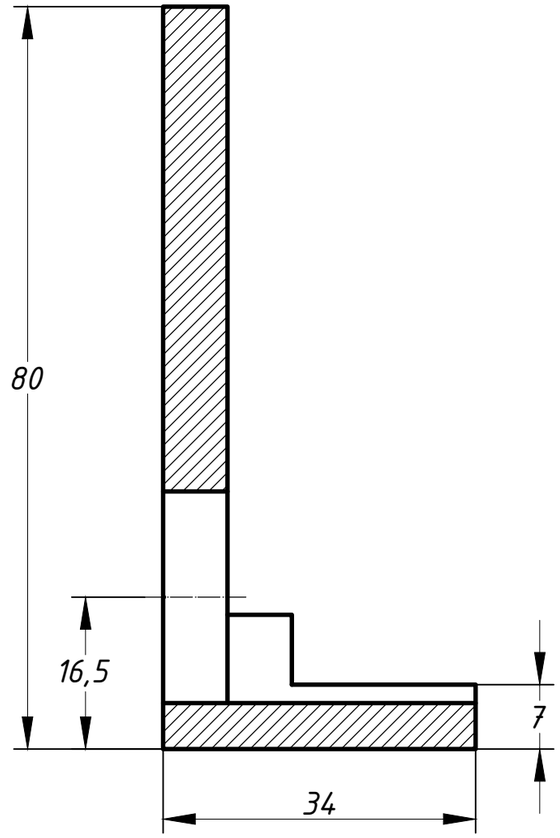
```
308.      delay(200);
309.    }
310.
311.    void abajo(){
312.      Serial1.println("Lapiz Abajo");
313.      myservo.write(PEN_DOWN);
314.      delay(200);
315.    }
```

3. Planos

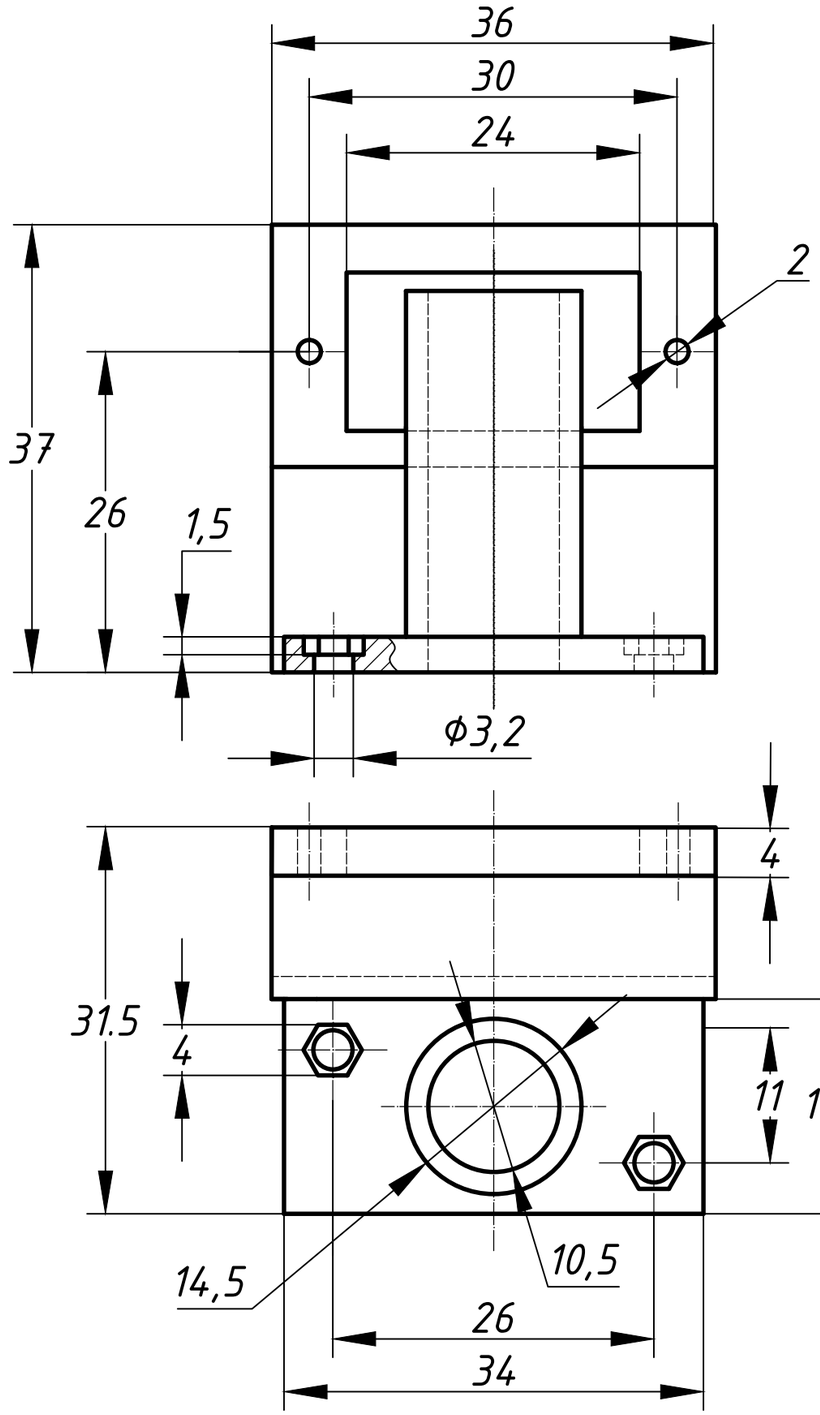
A continuación, se incluyen los planos de las piezas que han sido diseñadas específicamente para este robot.

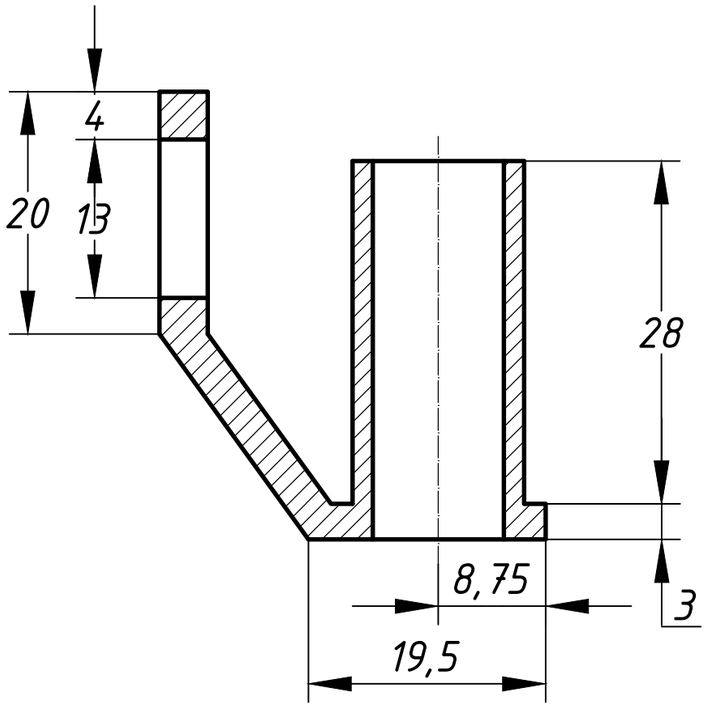
- Sujeción de los motores
- Sujeción del rotulador
- Sujeción de la bola
- Elevador del rotulador
- Sujeción de las ruedas
- Planchas cuerpo del robot



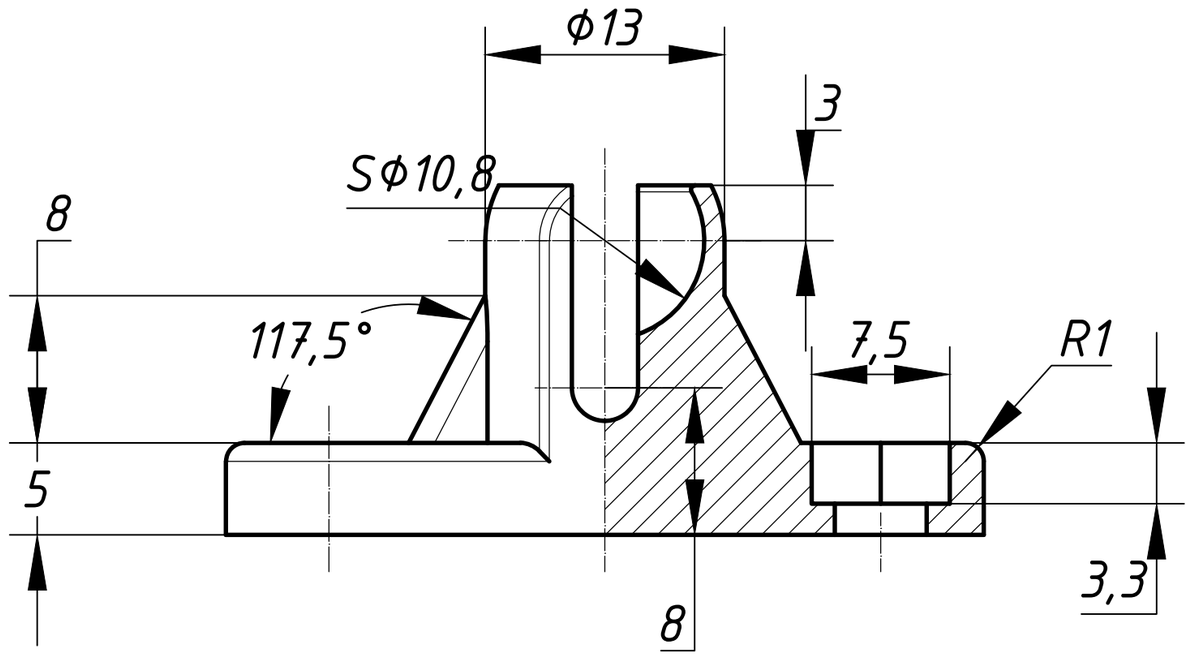


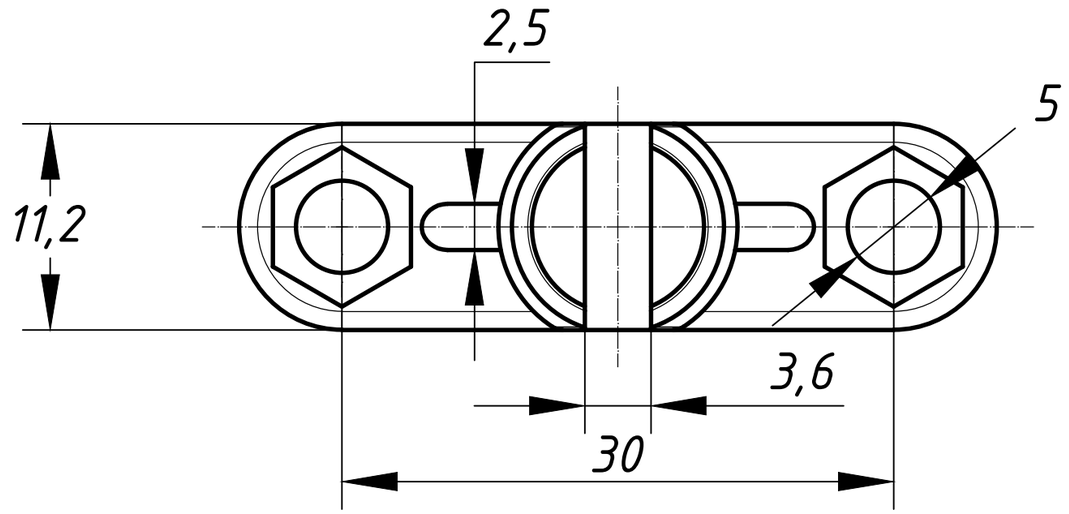
UNIVERSIDAD DE LEÓN ESCUELA INGENIERÍA INDUSTRIAL, INFORMÁTICA Y AEROESPACIAL	PROYECTO Diseño de un Robot 'Turtle', con control remoto, implementando el lenguaje de programación Logo		
ELEMENTO SUJECIÓN MOTORES	ESCALA 3:2	FECHA 24/05/2022	Nº PLANO 1
ALUMNO Óscar Conde Pérez	TRABAJO FIN DE GRADO		



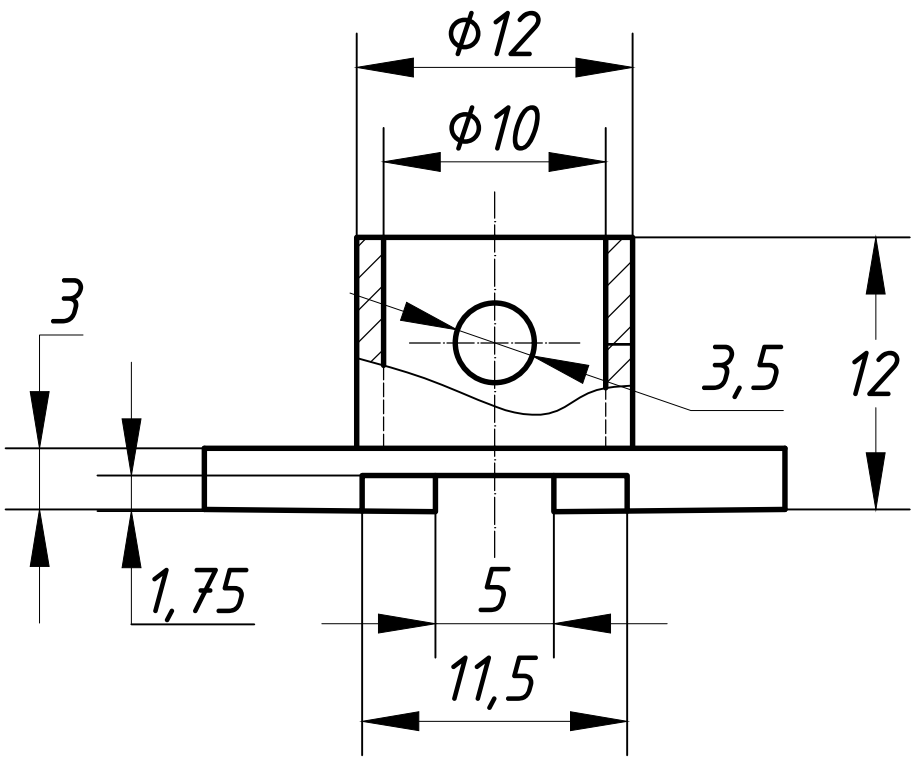


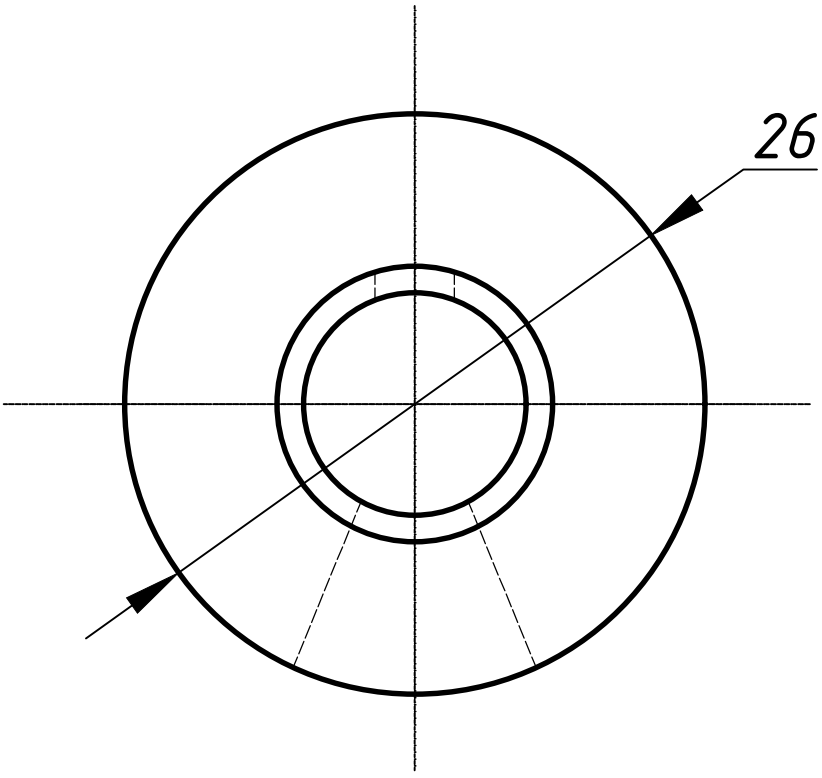
UNIVERSIDAD DE LEÓN ESCUELA INGENIERÍA INDUSTRIAL, INFORMÁTICA Y AEROESPACIAL	PROYECTO Diseño de un Robot 'Turtle', con control remoto, implementando el lenguaje de programación Logo		
ELEMENTO SUJECIÓN LÁPIZ DIBUJO	ESCALA 2:1	FECHA 24/05/2022	Nº PLANO 2
ALUMNO Óscar Conde Pérez	TRABAJO FIN DE GRADO		



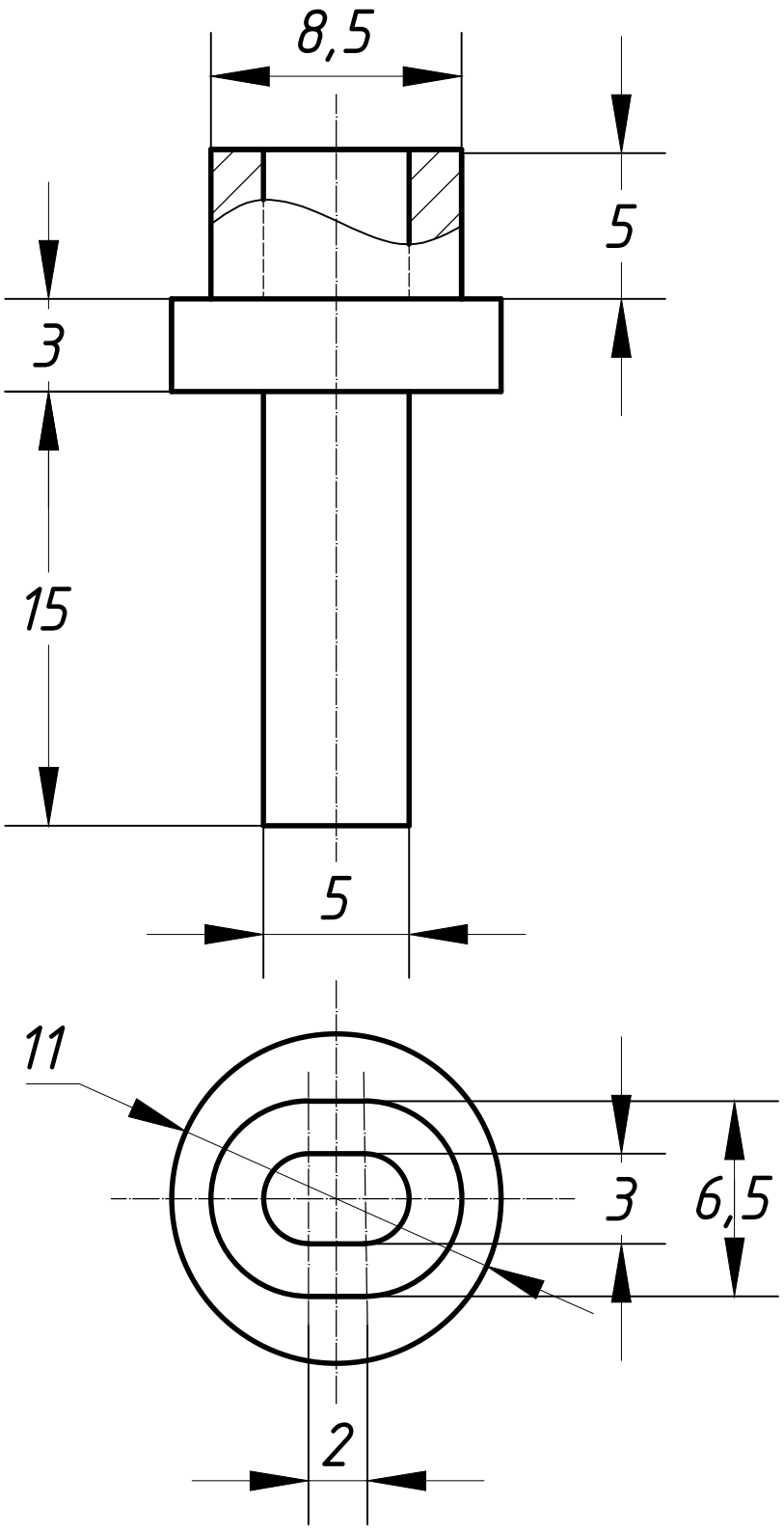


UNIVERSIDAD DE LEÓN ESCUELA INGENIERÍA INDUSTRIAL, INFORMÁTICA Y AEROESPACIAL		PROYECTO Diseño de un Robot 'Turtle', con control remoto, implementando el lenguaje de programación Logo	
ELEMENTO SUJECIÓN BOLA		ESCALA 3:1	FECHA 24/05/2022	Nº PLANO 3
ALUMNO Óscar Conde Pérez		TRABAJO FIN DE GRADO	

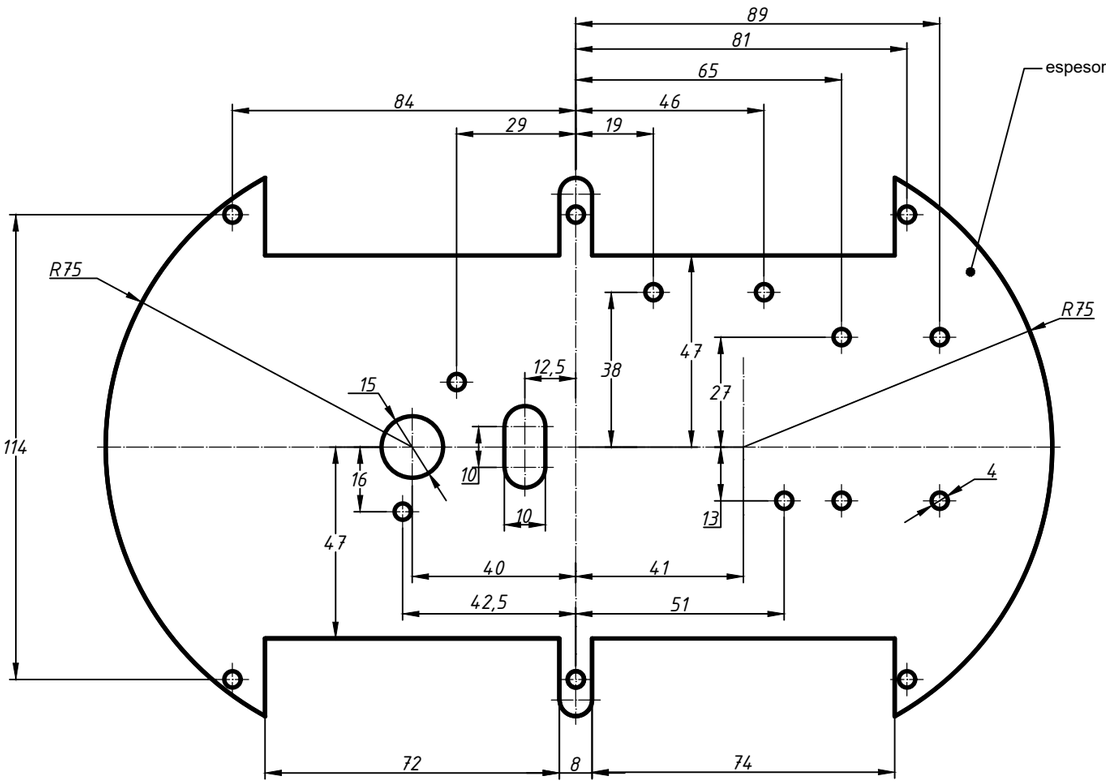




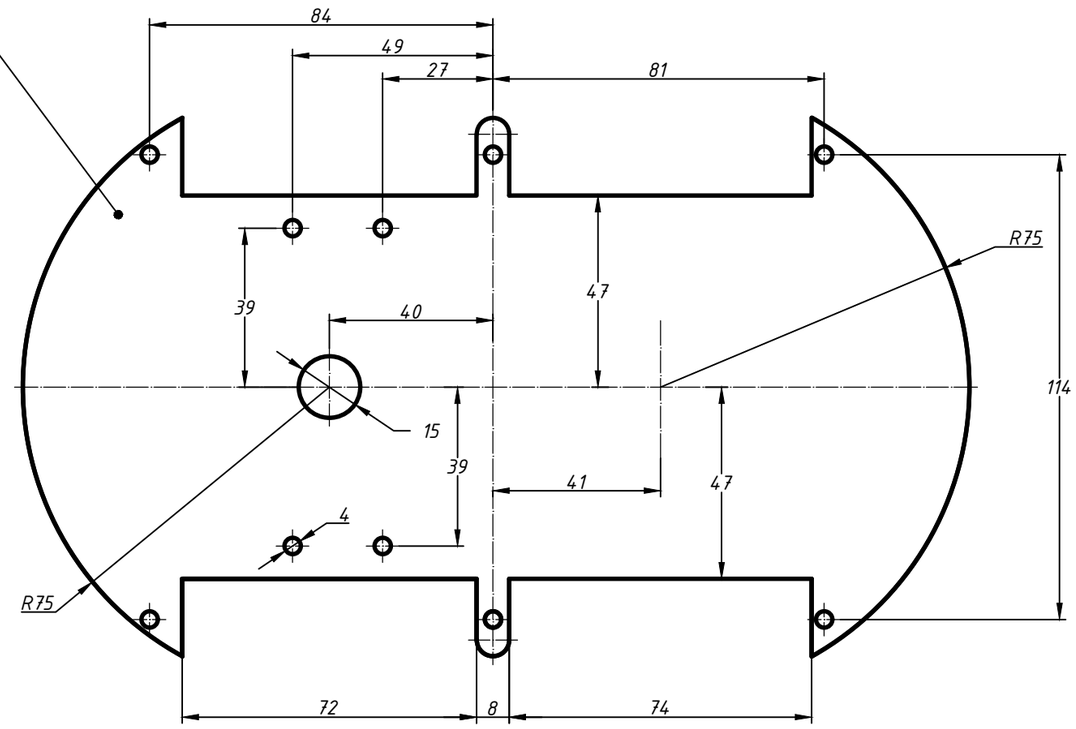
UNIVERSIDAD DE LEÓN ESCUELA INGENIERÍA INDUSTRIAL, INFORMÁTICA Y AEROESPACIAL	PROYECTO Diseño de un Robot 'Turtle', con control remoto, implementando el lenguaje de programación Logo		
ELEMENTO ELEVADOR LÁPIZ	ESCALA 3:1	FECHA 24/05/2022	Nº PLANO 4
ALUMNO Óscar Conde Pérez	TRABAJO FIN DE GRADO		



UNIVERSIDAD DE LEÓN ESCUELA INGENIERÍA INDUSTRIAL, INFORMÁTICA Y AEROESPACIAL	PROYECTO Diseño de un Robot 'Turtle', con control remoto, implementando el lenguaje de programación Logo		
ELEMENTO SUJECIÓN RUEDA	ESCALA 4:1	FECHA 24/05/2022	Nº PLANO 5
ALUMNO Óscar Conde Pérez	TRABAJO FIN DE GRADO		



PLACA SUPERIOR



PLACA INFERIOR

UNIVERSIDAD DE LEÓN ESCUELA INGENIERÍA INDUSTRIAL, INFORMÁTICA Y AEROESPACIAL	PROYECTO Diseño de un Robot 'Turtle', con control remoto, implementando el lenguaje de programación Logo		
ELEMENTO PLACAS SUPERIOR E INFERIOR ROBOT	ESCALA 2:3	FECHA 24/05/2022	Nº PLANO 6
ALUMNO Óscar Conde Pérez	TRABAJO FIN DE GRADO		

4. Pliego de condiciones

Las condiciones a seguir en el desarrollo de este proyecto no difieren de las de cualquier otro proyecto de las mismas características. Además de cumplir con la normativa indicada en el Reglamento electrotécnico de baja tensión. [50]

5. Presupuesto

En este apartado se hace referencia a los costes de producción del proyecto.

Coste mano de obra

Se incluye un desglose de las tareas, horas y precio la hora.

Tarea	Horas	€/hora	Coste final (€)
Análisis del sistema	50	20	1000
Elección actuadores	20	20	400
Elección microcontrolador	8	20	160
Programación	100	20	2000
Diseño soportes	20	20	400
Montaje final	30	20	600
Pruebas funcionamiento	10	20	200
		Total:	4760

Tabla 3 - 1 Coste mano de obra

El proyecto ha tenido una duración de 238h, el equivalente a 30 días laborales de jornadas de 8h.

Coste Software

En este apartado se incluye el software utilizado. Son programas necesarios para la realización del proyecto y que puede ser utilizado para otras actividades por lo que su coste no es total sino proporcional al tiempo que ha sido empleado.

Software	Precio licencia final	Precio licencia anual	Tiempo utilizado (meses)	Precio final (€)
Windows 10 pro	259 €		12 meses	18,13
AutoCad		2.342 €	1 mes	195,16
Microsoft 365 Personal		69 €	10 meses	57,5
Fusion 360		352 €	1 mes	29,3
			Total:	300,09

Tabla 3 - 2 Coste Software

Coste Final

A continuación, se realiza una suma de los costes parciales expuestos anteriormente y se añaden los costes indirectos y beneficios industriales.

Concepto	Coste (€)
Mano de obra	4760
Software	300,09
Costes indirectos (2%)	95,2
Beneficio industrial (10%)	476
Total	5631,29
IVA (21%)	1182,5709

Importe final	6813,8609

Tabla 3 - 3 Coste final

El importe final del proyecto de ingeniería es de seis mil ochocientos trece con ochenta y seis euros.

PRESUPUESTO DE PROTOTIPO

Coste de componentes

Componente	Modelo	Cantidad	Precio unitario (€)	Precio total (€)
Motor paso-a-paso	28BYJ-48	2	2,59	5,18
Circuito integrado	ULN2003	2	1,47	2,94
Placa protoboard	ELEGOO PCB mini	1	7,99	7,99
Cables	cable macho-hembra	1	3,5	3,5
Placa microcontrolador	Arduino Leonardo	1	22,55	22,55
Módulo comunicación	Kit comunicación Xbee	1	95,93	95,93
Servomotor	Microservo sg90	1	3,5	3,5
Cuerpo	Robot car kit smart v2.0	1	69,99	69,99
			Total:	211,58

Tabla 3 - 4 Coste componentes

Coste materias primas

Pieza	Material	Cantidad	Precio (€/kg)
Sujeción motores	PLA	2	20
Sujeción ruedas		2	
Elemento dibujo		1	
Sujeción bola		1	
		Total:	20

Tabla 3 - 5 Coste materiales

Coste Fabricación

Operario	Horas	€/hora	Coste final (€)
Impresión 3D	33,5	5	167,5
		Total:	167,5

Tabla 3 - 6 Coste fabricación

Coste final prototipo

Componentes	Mat.primas	Fabricación	Coste Final
211,58	20	167,5	399,08

Tabla 3 - 7 Coste final prototipo

Resumen

Presupuesto Ingeniería	6813,9
Presupuesto Prototipo	399,08
Presupuesto Total	7212,98

Tabla 3 - 8 Resumen costes

El presupuesto total del proyecto, sumando el presupuesto de ingeniería y el presupuesto del prototipo, es de siete mil doscientos doce con noventa y ocho euros.

6. Conclusiones finales

El objetivo principal de este proyecto era crear un dispositivo con las tecnologías actuales basado en el robot que desarrolló Seymour Papert a mediados de los años 70 para facilitar el estudio y comprensión de disciplinas como las matemáticas, el dibujo o la programación en un modelo educativo que el propio Seymour Papert definió como “construccionista”.

Con este proyecto se buscaba también adaptar el lenguaje de programación LOGO a los lenguajes actuales que pueden ser usados con los dispositivos de hoy en día.

Después de seleccionar las tecnologías actuales más óptimas y de solucionar las diferentes dificultades que se fueron encontrando a lo largo del desarrollo del proyecto (como se describe en el apartado de análisis de soluciones), se llegó al montaje actual del robot.

El montaje final del prototipo logra responder a todos los objetivos y objetivos secundarios que se plantearon al inicio de este proyecto. Consiguiendo un robot que se comunica de forma inalámbrica con otro dispositivo desde el que se mandan las órdenes de movimiento al robot. El movimiento del robot es suave, controlado y preciso. Permitiendo dibujar con gran exactitud cualquier forma geométrica regular que el usuario le indique.

Las ordenes que se le mandan al robot son las clásicas del robot original de la década de los 70 en lenguaje LOGO: adelante, retroceso, derecha o izquierda. Pero adaptadas al lenguaje de programación actual de Arduino. Además, se han incluido algunas funciones predeterminadas para dibujar formas geométricas. Estas funciones son: triángulo, cuadrado, círculo, hexágono y polígono. Qué hace cada una de estas funciones está explicado en profundidad en el apartado de resultados finales.

Una vez cumplidos todos los objetivos que se habían planteado para este proyecto y observando el funcionamiento y apariencia del montaje final, se plantean nuevos desarrollos o posibles mejoras para mejorar el prototipo.

- Buscar una mejor optimización del espacio del prototipo para reducir su tamaño y mejorar su robustez. Para ello habría que sustituir el controlador de Arduino por una placa PCB de menor tamaño y que permita aunar en un mismo espacio todos los componentes electrónicos del sistema.
- Diseñar una carcasa para recubrir el prototipo. Con esto cumplimos varios propósitos:

  - Proteger todos los componentes electrónicos de cualquier elemento externo que los estropé.
  - Proteger a los usuarios de tocar cables y componentes que les puedan causar algún tipo de daño.
  - Dar un mayor atractivo visual. Dando una forma de tortuga o algún animal a la carcasa, el robot será más atractivo para los niños y facilitará que quieran usarlo.

- Crear una interfaz gráfica más accesible para el usuario final. Ya que la interfaz del programa XCTU que permite la comunicación entre Xbee´s no es fácil de comprender para un usuario no familiarizado con el entorno.
- Crear una aplicación para cualquier dispositivo que permita controlar el robot y ver en pantalla lo que está dibujando el robot y como se está moviendo.
- Permitir al usuario escoger la forma en la que prefiere dar las órdenes al robot, ya sea escribiendo las instrucciones adelante, derecha, cuadrado, etc; Indicando las coordenadas donde quiere que se sitúe el robot, o dibujando en la aplicación mencionada en el punto anterior, la forma que se quiere representar.
- Introducir un control por voz que permita dirigir y controlar el robot.