TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO

AJUSTE EFICIENTE DE UN SISTEMA MEDIANTE EL CONTROLADOR PID DE ORDEN FRACCIONARIO

TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

PARA OBTENER EL TITULO PROFESIONAL DE:
INGENIERO EN MECATRÓNICA

PRESENTA:
FABRICIO OSWALDO BAUTISTA BAUTISTA
JOSÉ LUIS GARCÍA VÁZQUEZ

NOMBRE DEL ASESOR:
DRA. HAYDEE PATRICIA MARTÍNEZ HERNÁNDEZ

Instituto Tecnológico de Apizaco

APIZACO, TLAX., AGOSTO DE 2023

Apizaco, Tlaxcala, 02/Mayo/2023.

Circular No.: MM/226/2023.

ASUNTO: **Liberación de Proyecto para la Titulación Integral.**

MARTIN ROJAS RAMÍREZ
JEFE DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E

Por este medio informo que ha sido liberado el siguiente proyecto para Titulación integral:

Alumno:	FABRICIO OSWALDO BAUTISTA BAUTISTA
Carrera:	INGENIERÍA MECATRÓNICA
No. De Control:	16370808
Nombre del Proyecto:	AJUSTE EFICIENTE DE UN SISTEMA MEDIANTE EL CONTROLADOR PID DE ORDEN FRACCIONARIO.
Producto:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL.

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

ESTADOS UNIDOS MEXICANOS
SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DEPARTAMENTO DE METAL MECÁNICA

EFREN SÁNCHEZ FLORES
JEFE DEL DEPTO. DE METAL-MECÁNICA

HAYDEE PATRICIA MARTÍNEZ HERNÁNDEZ
ASESORA

C.p. Oficina de Titulación.-
C.p Departamento de servicios escolares
C.p. Expediente.- Jefe de proyectos de vinculación Metal-Mecánica

Apizaco, Tlaxcala, 02/Mayo/2023.

Circular No.: MM/255/2023.

ASUNTO: **Liberación de Proyecto para la Titulación Integral.**

MARTIN ROJAS RAMÍREZ
JEFE DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E

Por este medio informo que ha sido liberado el siguiente proyecto para Titulación integral:

Alumno:	JOSÉ LUIS GARCÍA VÁZQUEZ
Carrera:	INGENIERÍA MECATRÓNICA
No. De Control:	16370828
Nombre del Proyecto:	AJUSTE EFICIENTE DE UN SISTEMA MEDIANTE EL CONTROLADOR PID DE ORDEN FRACCIONARIO.
Producto:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL.

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DEPARTAMENTO DE METAL MECÁNICA

EFREN SÁNCHEZ FLORES
JEFE DEL DEPTO. DE METAL-MECÁNICA

HAYDEE PATRICIA MARTÍNEZ HERNÁNDEZ
ASESORA

C.p. Oficina de Titulación.-
C.p Departamento de servicios escolares
C.p. Expediente.- Jefe de proyectos de vinculación Metal-Mecánica

Apizaco, Tlax., 03/agosto/2023
No. OFICIO: DEP-CT/537/2023

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

FABRICIO OSWALDO BAUTISTA BAUTISTA
EGRESADO DE LA CARRERA DE INGENIERÍA MECATRÓNICA
CON NÚMERO DE CONTROL: 16370808
P R E S E N T E

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** cuyo tema es: **"AJUSTE EFICIENTE DE UN SISTEMA MEDIANTE EL CONTROL PID DE ORDEN FRACCIONARIO"**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga:

AUTORIZACIÓN DE IMPRESION

Sin otro particular por el momento, le envió un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DIVISIÓN DE ESTUDIOS PROFESIONALES

MARTÍN ROJAS RAMÍREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

c.p. archivo

Apizaco, Tlax., 11/agosto/2023
No. OFICIO: DEP-CT/543/2023

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

JOSE LUIS GARCIA VAZQUEZ
EGRESADO DE LA CARRERA DE INGENIERÍA MECATRÓNICA
CON NÚMERO DE CONTROL: 16370828
P R E S E N T E

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** cuyo tema es: **"AJUSTE EFICIENTE DE UN SISTEMA MEDIANTE EL CONTROL PID DE ORDEN FRACCIONARIO"**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga:

AUTORIZACIÓN DE IMPRESION

Sin otro particular por el momento, le envió un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DIVISIÓN DE ESTUDIOS PROFESIONALES

MARTÍN ROJAS RAMÍREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

c.p. archivo

AGRADECIMIENTOS

Primero que nada agradezco a Dios por darme la fuerza, el coraje y la inteligencia necesaria para culminar con esta etapa de mi vida. Por siempre escucharme, responder a mis plegarias y dame el ánimo necesario para siempre continuar, pero sobre todo, le agradezco permitirme terminar mi vida universitaria junto a toda mi familia.

Agradezco a mis padres por estar siempre al pie del cañón, procurando y apoyándome en cada momento aun cuando no lo merecía; A mi madre Ana, por el esfuerzo constante de día tras día para que saliera adelante, el amor y entera admiración hacia su valentía y coraje que tengo impregnada en cada fibra de mi cuerpo. A mi padre Sanson, por las pláticas, consejos, apoyo y el amor que siempre ha demostrado hacia mí desde pequeño, siempre ayudando a formarme como un hombre responsable, con carácter pero sin dejar de lado sus emociones.

A mis hermanos, Vladimir y Alexis, por todos los momentos que hemos tenido juntos, entre risas y peleas, triunfos y fracasos, siempre han estado ahí de formas distintas, siempre para mí.

A mis amigos de la universidad que más que amigos son mi familia, Kenia Osorno, Emmanuel Corona, Giovanni Fragoso, Erick J. López, José Luis García, Verónica Sánchez, Giovani Cuevas sin ellos la universidad nunca habría sido lo mismo.

De igual forma quiero agradecer a cada una de las personas que he conocido a través de los pocos años que tengo de vida, todos y cada uno de ustedes han ayudado y siguen ayudando a formarme como el ser humano del que estoy orgulloso de ser y es una de las cosas que más atesoro en la vida, infinitas gracias.

Fabricio Oswaldo Bautista Bautista

A Dios, por ayudarme a terminar mis estudios universitarios, gracias por darme la capacidad intelectual, la humildad, el ahínco y ánimo para concluir esta meta.

A mis padres, Lubin y María Clara. No existen palabras suficientes para expresar la profundidad de mi gratitud por todo lo que han hecho por mí, por su sacrificio. Su inquebrantable dedicación y apoyo me han brindado la fortaleza necesaria para concluir esta etapa.

A pesar de que hubo dificultades y desafíe normas y valores que me enseñaron nunca dejaron de apoyarme y mostrarme su cariño. Su constante guía me ayudó a convertirme en la persona que soy y estoy muy agradecido por ello. Valoraré para siempre su amor incondicional y su compromiso de estar a mi lado sin importar las circunstancias.

Ustedes son todo mi mundo, lo más profundo. Gracias por todo ese mal que hicieron humo, fue nuestro triunfo. Hoy sigo parado, ustedes son un milagro y yo un diluvio. Este logro marca un hito en la historia de mi vida, este logro es de ustedes.

A mis hermanos, César David y Lubin. Quiero expresar mi más profundo agradecimiento por ser ejemplos de vida tan inspiradores para mí. Su determinación, bondad y fortaleza son una constante fuente de motivación en mi camino. Su dedicación y perseverancia en superar los desafíos de la vida son verdaderamente admirables. Gracias por inspirarme a creer en mis propias capacidades y a nunca renunciar. Siempre llevaré conmigo su impacto positivo en mi vida.

A todos los Amigos, Familiares y Maestros que he me han apoyado o brindado un consejo a lo largo de mi vida. Muchas gracias.

José Luis García Vázquez

Por último y no menos importante le queremos expresar nuestro más sincero agradecimiento a la Dra. Haydee Patricia Martínez Hernández por su valioso apoyo y orientación durante el proceso de desarrollo de este proyecto.

Su experiencia, paciencia y dedicación fueron fundamentales para que pudiéramos completar este proyecto con éxito. Cada consejo y dirección que nos brindó contribuyeron significativamente a nuestro crecimiento académico y personal. Estamos verdaderamente agradecidos por su compromiso, humanismo y humildad, así como por brindarnos la confianza necesaria para llevar este proyecto avante.

Su guía ha sido fundamental para nuestro crecimiento y aprendizaje, toda la vida le estaremos profundamente agradecidos por tenerla como nuestra asesora. Dios la bendiga, Doctora.

Fabricio Oswaldo Bautista Bautista

José Luis García Vázquez

CONTENIDO

RESUMEN

Este proyecto consiste en la implementación de un sistema de control PID (Proporcional, Integral y Derivativo) de Orden Fraccionario en un robot seguidor de línea, esto es un dispositivo que sea capaz de detectar y seguir una línea, la cual se encuentra ubicada en el suelo de una superficie.

Para la integración del cálculo fraccionario en un robot seguidor de línea se propone el uso de control automatizado, que auxiliado por el uso de herramientas computacionales da como resultado el uso de sensores.

Lo más remarcable es poder integrar el cálculo fraccionario a un controlador proporcional integral derivativo, a partir de eso surgen los siguientes resultados específicos:

I. Seguir la línea, haciendo que tenga una mejor coordinación al momento de ejecutar el programa.

II. Comunicación y transferencia de datos de forma inalámbrica entre el circuito de detección de línea ubicada en el robot y circuito de control por computador.

III. Aumentar su velocidad de manera que esta sea superior al PID normal.

IV. Estabilizar el sistema PID.

V. Tener un mejor agarre en las curvas a una mayor velocidad.

VI. Analizar el comportamiento de un PID comparado con el PID de OF, y hacer una analogía de las diferencias y si son significativas o despreciable.

CAPITULO I. ESTADO DEL ARTE

En los últimos años ha adquirido una especial importancia de divulgación e implementación de un robot seguidor de línea. La información sobre sus componentes, la matemática que se ve reflejada en ellos, la programación que necesitan, y la teoría de control que tiene integrada, así como los alcances que este tipo de tecnologías pueden tener sea de manera educativa o industrial.

A continuación, algunos antecedentes relacionados al tema de investigación realizados en México que tienen alcance fuera del país. Esto se divide en dos:

a) Robot seguidor de línea más veloz y preciso.

b) Control autónomo de un robot seguidor de línea utilizando un controlador Q- Learning

1.1 Robot seguidor de línea más veloz y preciso.

Alumnos del Instituto Politécnico Nacional (IPN) se dieron a la tarea de crear un prototipo para mejorar el robot seguidor de líneas, con el fin de optimizar su funcionamiento y hacerlo más veloz, ligero y preciso.

En un comunicado, se informó que los componentes básicos de este tipo de robot son los motores, sensores, ruedas, baterías y tarjetas de control, pero para mejorarlo los estudiantes le añadieron un motorreductor, turbinas y sensores.

Los alumnos del Centro de Estudios Científicos y Tecnológicos (CECyT) 2 Miguel Bernard, explicaron que la mayoría de los seguidores de línea son construidos por motorreductores muy pesados, volviéndolos más lentos, también tienen una tabla con cinco o seis sensores que en ocasiones no detectan con precisión los movimientos.

Lisania Amador Tapia, Luis Fernando Haro Campos, Jorge Eduardo Correa Argumosa y Maximiliano Ducoing Espinoza denominaron a su robot como “Seguidor de Líneas Avanzado” porque cuenta con un motorreductor de diez a uno que lo hace más ligero y más veloz.

Además le adaptaron una barra con 16 sensores y una turbina para que el androide esté lo más pegado al piso y tenga más tracción.

Los politécnicos mencionaron que la función de los sensores es enviar al microprocesador las señales de entrada y salida para que las traduzca y luego las mande a los drivers, que ordenan a los motores si aumentan, disminuyen o mantienen la velocidad.

El robot es controlado por conexión bluetooth, a través de una computadora, y cuenta con una tarjeta madre.

Ducoing Espinoza señaló que no utilizaron una tabla fenólica porque es muy grande y lo que quería era hacerlo más ligero.

El prototipo también tiene componentes como los motores, encargados de que las llantas giren; éstas están hechas de una combinación de silicón y caucho para que se adhieran mejor.

Además una pila recargable de litio de 7.4 volts y 3 ampers; tres botones pulsadores que sirven para activar la turbina, calibrar al robot y pueda distinguir entre la línea y el fondo y prender al robot.

1.2 Control autónomo de un robot seguidor de línea utilizando un controlador Q- Learning

El controlador convencional para este tipo de robots es el controlador proporcional (P). Teniendo en cuenta las características mecánicas desconocidas del robot y las incertidumbres como la fricción y las superficies resbaladizas, el modelado del sistema y el diseño del controlador pueden ser extremadamente desafiantes. Se presenta el modelado matemático del robot y se diseña un simulador basado en este modelo.

Los métodos básicos de Q-learning se basan en la explotación pura y los métodos ε-voraz, que ayudan a la exploración, pueden dañar el rendimiento del controlador después de completar el aprendizaje al explorar acciones no óptimas.

El método Q-learning basado en recocido simulado aborda este inconveniente al disminuir la tasa de exploración cuando aumenta el aprendizaje. La simulación y los resultados experimentales se proporcionan para evaluar la eficacia del controlador propuesto.

CAPÍTULO II. CÁLCULO FRACCIONARIO

El Cálculo fraccionario es la denominación acuñada para la extensión del cálculo que permite considerar la integración y la derivación de cualquier orden, no necesariamente entero. En el dominio del tiempo, los operadores derivada e integral fraccionarios vienen definidos por la operación de convolución, por lo que están especialmente indicados para describir fenómenos de memoria.

2.1 Orígenes del cálculo fraccionario

La primera mención a la posibilidad de extender el sentido de la expresión para $\frac{d^n y}{dx^n}$ es n no entero se encuentra en la correspondencia entre Leibnitz y L*Hópital, y hasta el siglo XIX fue un asunto que sólo trataron algunos eminentes científicos, como Euler, Laplace, Fourier, Liou-ville, Riemann o Abel, siendo este último quien por primera vez lo aplicó en Física al solucionar una ecuación integral surgida en la formulación del llamado problema de la tautócrona. Á partir de entonces, y con especial énfasis en las últimas cuatro décadas, el cálculo fraccionario se ha empleado con éxito en el modelado de fenómenos y sistemas físicos estudiados en multitud de campos de la ciencia y la ingeniería. Entre ellos se pueden destacar la ciencia de materiales, la teoría del caos y los fractales, la electrónica de dispositivos, la física teórica, la mecánica, etc...

En cierta forma, las aplicaciones del cálculo fraccionario han experimentado una evolución análoga a la experimentada por la propia teoría de control, siguiendo dos caminos separados según que el punto de partida fuera el dominio del tiempo o el de la frecuencia. Mientras que las aplicaciones del cálculo fraccionario al modelado de sistemas físicos han utilizado, salvo en el caso de algunas aplicaciones en electroquímica, el dominio del tiempo, las aplicaciones en control han utilizado, mayoritariamente y desde el principio, el dominio de la frecuencia.

Las primeras aplicaciones del cálculo fraccionario en control se dieron a principios de los años 60. Estas primeras aplicaciones hacían uso del operador integral de orden no entero para el control de servos y de sistemas con saturación. Estos trabajos, probablemente sin conocimiento de los autores, hacían uso de las propiedades de la que Bode denominó función de transferencia ideal en lazo abierto, cuya forma era, $F(s) = \frac{A}{s^{\propto}}$ es decir, era un integrador fraccionario. La principal propiedad de esta función es que, cuando se conecta en lazo cerrado, da sistemas robustos a cambios en la carga o ganancia del proceso, es decir, los cambios en la carga se reflejan en cambios en el ancho de banda del sistema en lazo cerrado, pero se mantiene constante el margen de fase o, lo que es equivalente, la sobreoscilación máxima.

Esta propiedad es la que marcó las aplicaciones en control que a partir de los años 70 se han venido desarrollando en la Universidad de Burdeos dando como resultado el sistema CRONE (Control Robusto de Orden No Entero). A partir de entonces, aunque de forma aislada, han venido apareciendo trabajos que aplicaban los operadores fraccionarios a problemas de control, tanto explotando su robustez y su capacidad para tratar el control de sistemas de orden fraccionario, como para el control de sistemas de parámetros distribuidos. Así mismo, han aparecido artículos sobre aspectos más generales del uso de controladores fraccionarios, o sobre el uso del cálculo fraccionario en disciplinas íntimamente relacionadas con la teoría de control, como son el tratamiento de señales, el ajuste de curvas y la estimación de parámetros, o el diseño y realización de filtros.

2.2 Fundamentos de cálculo fraccionario

2.2.1 Operadores Fraccionarios

La integral fraccionaria De acuerdo con la concepción de Riemann-Liouville, la noción de integral fraccionaria de orden α, Re (α) > 0, es una consecuencia natural de la fórmula atribuida a Cauchy, que reduce el cálculo de la primitiva correspondiente a la integración de multiplicidad n de una función f(t) a una integración simple de tipo convolución. La fórmula de Cauchy puede expresarse como:

$$I_c^n f(t) \equiv \frac{1}{(n-1)!}\int_c^t t(-\tau)^{n-1} f(\tau) d\tau$$

$$\tau > C, n \in \mathbb{Z}^+$$

Donde ℤ+ es el conjunto de los números enteros positivos. De una forma natural, se puede extender la validez de la fórmula anterior de valores del índice enteros positivos a valores reales positivos utilizando la función Gamma. Teniendo en cuenta que (n − 1)! = Γ(n), e introduciendo el número real positivo α, se define la integral fraccionaria de orden α ∈ ℝ+ como.

Donde ℝ+es el conjunto de los números reales positivos. Su transformada de Laplace viene dada por la expresión:

$$£\{I\alpha f(t)\} = s - \alpha\, F(s)$$

La derivada fraccionaria Llamando Dn con $n \in \mathbb{Z}^+$ al operador derivada de orden n, e introduciendo el entero positivo m tal que m - 1 < α < m, se obtiene la definición de Riemann-Liouville para la derivada fraccionaria de orden α ∈ ℝ+.

$$D^{\propto} f(t) \equiv \left[\frac{d^m}{dt^m}\left[\frac{1}{\Gamma(\mathrm{m}-\alpha}\right]\int_0^t \frac{f(\tau)}{(t-\tau)^{\alpha-m-1}}\,d\tau\right]$$

$$m\ -\ 1\ <\ \alpha\ <\ m, m\ \in\ \mathbb{Z}+$$

Cuya transformada de Laplace puede expresarse como:

$$£[D\alpha f(t)]\ =\ s\alpha F(s)\ -\ \sum_{k=0}^{m-1} skf\alpha - k - 1\ (0)$$

Siendo (0) el valor de la derivada de orden 1 de la función f en el origen. Una definición alternativa de la derivada fraccionaria es la introducida por Caputo.

$$D\alpha f(t) \equiv \frac{1}{\Gamma(m-\alpha)}\int_0^t \frac{fm(\tau)}{(t-\tau)\alpha - m - 1}\,d\tau$$

$$m\ -\ 1\ <\ \alpha\ <\ m, m\ \in\ N+$$

Esta definición incorpora los valores iniciales de la función y sus derivadas de orden entero menor, es decir, condiciones iniciales que son físicamente interpretables de la manera tradicional. Así, su transformada de Laplace es de la forma:

$$£[D\alpha f(t)] =\ s\alpha F(s) - \sum_{k=0}^{m-1} skf\alpha - k - 1\ (0)$$

De gran interés para la implantación discreta de controladores y filtros de orden fraccionario es la definición de Griindwald-Letnikov para la derivada fraccionaria. Dicha definición se basa en la generalización de la bien conocida fórmula de las diferencias de orden $n \in \mathbb{Z}+$, para el caso de $\alpha \in \mathbb{R}+$, y supone el uso de las diferencias regresivas, es decir:

$$D^{\propto} f(t)t = kh \lim_{h \to 0} h \sum_{j=0}^{k} (-1)^j \binom{\propto}{j} f(kh - jh)$$

Siendo su transformada de Laplace de la forma:

$$£\{D\alpha\, f(t)\} \;=\; s\alpha F(s)$$

Se sabe que los problemas clásicos de relajación y oscilación vienen gobernados por ecuaciones diferenciales lineales ordinarias de orden uno y dos respectivamente. En esta sección se estudiarán las ecuaciones diferenciales que resultan de generalizar el orden. Así, se puede reformular el problema considerando la siguiente expresión general para la ecuación diferencial fraccionaria de orden $\alpha > 0$.

$$D\alpha\, u(t) \;+\; u(t) \;=\; D\alpha \left(u(t) - \sum_{k=0}^{m-1} \frac{t^k}{k!} u^k(0^+) \right)$$

$$+u(t) \;=\; q(t), t \;>\; 0,$$

Siendo m un entero positivo definido por $m - 1 < \alpha < m$, que determina el número de valores iniciales $(0+) = bk$, $k = 1, 2, \ldots m - 1$. Los casos particulares de relajación y oscilación se obtienen haciendo $m = 1$ y $m = 2$, respectivamente.

Introduciendo la función de Mittag-Lefler, $(-t\alpha)$, definida mediante la expresión:

$$E\alpha(z) \;=\; \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(ak + 1)}$$

Se obtiene la solución en la forma:

$$u(t) = \sum_{k=0}^{m-1} bk\ IkE\alpha(-t\alpha) - \int_0^t q(t - r)\ E'\alpha(-r\alpha)dr$$

Como se puede ver, la función de Mittag-Leffle desempeña en este tipo de ecuaciones diferenciales un papel análogo al que desempeña la función exponencial en las ecuaciones diferenciales ordinarias. En efecto:

- Cuando α no es entero, es decir, para $m - 1 < \alpha < m$, $m - 1$ representa la parte entera de α ($m - 1 \equiv [\alpha]$) y m el número de condiciones iniciales necesarias y suficientes para asegurar la unicidad de la solución, (t);

- Las m funciones $IkE\alpha(-t\alpha)$, donde $k = 0, 1 \ldots, m - 1$, son las soluciones particulares de la ecuación homogénea que satisfacen las condiciones iniciales, y, por ello, son las soluciones fundamentales de la ecuación diferencial fraccionaria

- La función $E'_{\propto}(-t^{\propto})$, , que es la primera derivada de la función $E'_{\propto}(-t^{\propto})$,, es la respuesta impulsiva.

Es claro que para conocer la forma de la solución es preciso conocer ciertas propiedades de la función básica $E_{\propto}(-t^{\propto})$. Dicha función empieza por tener una evolución correspondiente a una relajación anómala ($\alpha < 1$), pasa por una evolución exponencial ($\alpha = 1$) y alcanza una oscilatoria ($1 < \alpha < 2$) que se hace no amortiguada para $\alpha = 2$.

2.3 Sistemas dinámicos de orden fraccionario

Una vez establecidas las definiciones fundamentales del cálculo fraccionario y determinados los tipos de soluciones de las ecuaciones diferenciales de orden fraccionario. Dicho análisis, tal como es habitual para los sistemas de orden entero, partirá de los modelos o representaciones de dichos

sistemas en los diferentes dominios (tiempo, Laplace y Z) para el estudio de su comportamiento, tanto en régimen transitorio como en régimen estacionario, discutiendo las condiciones y criterios de estabilidad, controlabilidad y observabilidad.

2.*3.1Modelos y Representaciones*

Ecuaciones diferenciales Partiendo de lo establecido en la sección anterior, se pueden formular las ecuaciones constitutivas de un sistema dinámico de orden fraccionario, lineal, monovariable e invariante en el tiempo, de la forma:

$$\sum_{k=0}^{m} ak\ D\alpha k y(t) = \sum_{k=0}^{m} bk\ D\beta k u(t)$$

En la ecuación anterior pueden considerarse dos casos particulares que dan lugar a dos tipos de sistemas de gran interés: los sistemas de orden conmensurable y los sistemas de orden racional.

Se definirán estos sistemas como sigue:

Definición 1. Un sistema es de orden conmensurable si queda descrito por una ecuación diferencial donde todos los órdenes de derivación son múltiplos enteros de un orden base, a. Es decir, sistemas para los que se cumple que:

$$\alpha k, \beta k\ =\ k\alpha, \alpha\ \in\ \mathbb{R}, k\ \in\ \mathbb{Z}$$

Así, la ecuación diferencial se puede poner de la forma:

$$\sum_{k=0}^{n} ak\ Dk\alpha y(t) = \sum_{k=0}^{m} bk\ Dk\alpha u(t)$$

Definición 2. Un sistema es de orden racional si es de orden conmensurable y además se cumple que: $\propto = \frac{1}{q}, q \in \mathbb{Z}^+$

Utilizando la definición de Griindwald- Letnikov para la derivada fraccionaria, se pueden obtener modelos discretos de los sistemas fraccionarios en forma de ecuaciones en diferencias generalizadas.

Representaciones entrada-salida Si aplicamos la transformada de Laplace con condiciones iniciales nulas, o la transformada Z a la ecuación en diferencias equivalente, se pueden obtener las representaciones entrada-salida (externas) de los sistemas fraccionarios. En el caso de los modelos continuos, un sistema fraccionario vendrá representado por una función de transferencia de la forma:

$$G(z) = \frac{Y(s)}{U(s)} = \frac{\Sigma_{k=0}^{m} b_k s^{\beta k}}{\Sigma_{k=0}^{m} \alpha_k s^{\alpha k}}$$

Para el caso de sistemas discretos se obtiene una función de transferencia discreta de la forma:

$$G(z) = \frac{\Sigma_{k=0}^{m} b_k (w(z^{-1}))^{\beta k}}{\Sigma_{k=0}^{m} \alpha_k (w(z^{-1}))^{\alpha k}}$$

Donde $(w(z^{-1})$ es la transformada Z del operador Δ_h^1, en otras palabras, el equivalente discreto del operador de Laplace s.

Como puede verse en las expresiones anteriores, un sistema de orden fraccionario tiene una función de transferencia no racional en el dominio de Laplace, o, en el dominio de Z, una función de transferencia discreta de orden ilimitado respecto de z, ya que sólo en el caso de $a_k \in \mathbb{Z}$ habrá un número limitado de coeficientes $(-1)^l \binom{a_k}{l}$ distintas de cero. En vista de ello, se puede decir

que un sistema de orden fraccionario tiene una memoria ilimitada, siendo los sistemas de orden entero un caso particular de los mismos.

En el caso de que el sistema sea de orden conmensurable, la función de transferencia continua tendrá la forma:

$$G(s) = \frac{\Sigma_{k=0}^{m} b_k (s^a)^k}{\Sigma_{k=0}^{m} \alpha_k (s^a)^k}$$

2.3.2Representaciones de estado

En el caso general de sistemas multivariables, puede obtenerse una representación espacio-estado de la forma:

$$D^a \boldsymbol{x} = \boldsymbol{Ax} + \boldsymbol{Bu}$$

$$\mathbf{y = Cx + Du}$$

Donde: $\alpha = [\alpha_0\, \alpha_1\, \alpha_2\, \ldots\, \alpha_n]$, $\boldsymbol{u} \in \mathbb{R}^l$ es el vector de entradas $\boldsymbol{u} \in \mathbb{R}^n$, es el vector de estados, $\boldsymbol{u} \in \mathbb{R}^l$ es el vector de salidas, $\boldsymbol{A} \in \mathbb{R}^{nxn}$ es la matriz de estado, $\boldsymbol{B} \in \mathbb{R}^{nxn}$ es la matriz de entrada, $\boldsymbol{C} \in \mathbb{R}^{nxl}$ es la matriz de salida, $\boldsymbol{D} \in \mathbb{R}^{pxl}$ es la matriz de transmisión directa.

Para el caso particular de sistemas de orden conmensurable, se pueden obtener representaciones de estado equivalentes a las usuales para sistemas de orden entero, sin más que utilizar el operador.

D^a, y definir $D^a x_k = x_{k+1}, k = 1,2, \ldots n-1$.

Uno de los problemas que plantea la representación de estados de los sistemas fraccionarios es la falta de una definición físicamente consistente de las variables de estado. S1 bien hay representaciones de estado para los sistemas de orden entero en las cuales las variables de estado no tienen correspondencia con magnitudes físicas definidas y medibles, cualquiera de ellas puede

reducirse a otra en la cual sí la tengan. Así, vemos que el problema de la interpretación de las variables de estado está relacionado con la posibilidad de medir las magnitudes físicas asociadas a las variables de estado, y hasta el momento no se han dado soluciones satisfactorias para los sistemas fraccionarios. Lo mismo podríamos decir para el caso de las condiciones iniciales.

Si se tiene un sistema mono variable de orden conmensurable, cuya representación de estado es de la forma

$$D^{\alpha}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathrm{u}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathrm{u}$$

y se toman transformadas de Laplace, suponiendo que se utiliza la definición de Caputo para la derivada fraccionaria, se obtiene, en el caso de condiciones iniciales nulas, la función de transferencia del sistema como:

$$G(s) = \boldsymbol{C}(s^{\boldsymbol{a}}\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}$$

Por otra parte, definiendo:

$$\Phi(t) \equiv \{(s^{\boldsymbol{a}}\boldsymbol{I} - \boldsymbol{A})^{-1}s^{a-1}\}, t \geq 0,$$

Y aplicando las propiedades de la transformada de Laplace, los estados se pueden obtener a partir de la expresión:

$$(t) = (t)(0) + D^{1-\alpha}\{\Phi(t) * [\mathbf{B}u(t)]\}$$
$$= \Phi(t)x(0) + \int_0^t D^{1-a}\{\{\Phi(\mathrm{t}-\tau)\}\boldsymbol{B}u(\tau)d(\tau)$$

Como se puede ver, (t) es la matriz que habitualmente se conoce como matriz de transición de estados. Siguiendo un procedimiento análogo al utilizado para los sistemas lineales de orden entero,

se puede determinar la forma de la matriz de transición de estados. Utilizando la definición de Caputo para la derivada fraccionaria, la solución puede expresarse como:

$$x(t) = \left(\sum \frac{A^k t^{ka}}{\Gamma(1+ka)}\right) x(0)$$

$$= E\alpha(\boldsymbol{A}t^a)x(0) \;=\; \Phi(t)\, x(0)$$

Es claro pues que la función de Mittag-Leffler desempeña el mismo papel para este tipo de sistemas que el desempeñado por la función exponencial para los sistemas de orden entero. La bien conocida matriz exponencial, e^{at}, no es más que un caso particular de la matriz exponencial generalizada $E_a(\boldsymbol{A}t^a)$, a la que podríamos llamar función matricial de Mittag- Leffter.

2.3.3 Controlabilidad y Observabilidad Para Sistemas de Orden Conmensurable

En Teoría Moderna de Control (la basada en la representación de estados) dos conceptos fundamentales, tanto para el análisis de sistemas como para el diseño de controladores, son los de controlabilidad y observabilidad. Adoptando las definiciones habituales para estos conceptos, se puede demostrar que para que el sistema sea controlable debe existir solución única para la ecuación matricial

$$(t_0) = \delta b,$$

Donde:

$$b_i = \int_0^{t_f} a_i((t_0 - \tau)^a u(\tau) d\tau$$

$$\mathbf{b} = [b_1\ b_2\ b_3 \ldots b_{n-1}]^T$$

$$\xi = [\mathbf{B}\ \mathbf{AB}\ \ldots\ldots\ \mathbf{A}^{n-2}\ \mathbf{B}\ \mathbf{A}^{n-1}\mathbf{B}]$$

Para que la ecuación tenga una solución única, ha de cumplirse que el rango de ξ, la matriz de controlabilidad, sea n. Así pues, es claro que las condiciones de controlabilidad para un sistema de orden conmensurable son las mismas que para un sistema de orden entero, sin más que construir su descripción de estado en base al operador D^a o, equivalentemente, $\lambda = s^a$. Lo mismo se puede decir en cuanto a las condiciones de observabilidad.

2.4 Estabilidad

De una manera general, el estudio de la estabilidad de los sistemas fraccionarios puede realizarse estudiando las soluciones de las ecuaciones diferenciales que los caracterizan, y se ha visto que la función de Mittag-Lefñer representa la solución fundamental, dependiendo la forma de las soluciones de los argumentos y parámetros de esta función, o, en otras palabras, de los coeficientes y órdenes de derivación de la ecuación diferencial fraccionaria.

Una manera alternativa es partir de la función de transferencia del sistema. Para hacer este estudio, es preciso recordar que una función del tipo a la transformación $w = s^{1/3}$

$$(s) = a_n s^{\alpha} n + a_{n-1} s^{\alpha} n-1 + \ldots a_0 s^{\alpha} 0,$$

Con $a_i \in \mathbb{R}^+$, es una función multivaluada de la variable compleja s cuyo dominio puede verse como una superficie de Riemann de un número de hojas que será finito sólo en el caso de que $\forall i$, $a_i \in Q^+$,, siendo la hoja principal la definida por $-\pi < \arg(s) < \pi$. $\forall i$, $a_i \in Q^+$, es decir, $\alpha = 1/q$, q entero positivo, las q hojas de la superficie de Riemann correspondiente vienen determinadas por:

$$s = |s| e^{j\emptyset}, (2k +)\pi < \emptyset < (2k + 3)\pi,$$

$$k = -1, 0, \ldots q - 2$$

2.4.1 Condiciones de estabilidad

Hechas las consideraciones previas del apartado anterior, se pueden establecer las condiciones de estabilidad de los sistemas fraccionarios.

En el caso más general, se puede decir que un sistema fraccionario con función de transferencia no racional, $(s) = (s)/(s)$, es estable para entrada y salida limitadas (BIBO estable) si y sólo si se cumple la siguiente condición:

$$\exists \mathrm{M}, |(s)| \leq M, \forall s/Re(s) \geq 0$$

La condición anterior se cumplirá si todas las raíces de la función (s) que están localizadas en la hoja principal de Riemann y no son raíces de (s) tienen parte real negativa.

Para los sistemas de orden conmensurable, que tienen una ecuación característica que es un polinomio en la variable compleja $\lambda = s^a$, la condición de estabilidad se puede expresar como:

$$|\arg(\lambda_i)| > a\frac{\pi}{2}$$

Siendo λ_i las raíces del polinomio característico. Para el caso particular de $\alpha = 1$ se obtiene la condición de estabilidad conocida para los sistemas lineales.

$$|\arg(\lambda_i)| > a\frac{\pi}{2}, \forall \lambda_i / Q(\lambda_i) = 0$$

2.4.2 Criterios de estabilidad

A la vista de todo lo expuesto anteriormente se puede establecer lo siguiente:

- El hecho de que todos los coeficientes de la ecuación característica sean del mismo signo no es condición necesaria ni suficiente para garantizar la estabilidad.
- El número de raíces de la ecuación característica en la hoja principal de Riemann no depende de la mayor potencia de la variable compleja en la ecuación característica.

En la actualidad no se dispone de técnicas polinómicas, tipo Routh o Jury, para analizar la estabilidad de los sistemas fraccionarios. Sólo las técnicas geométricas de análisis complejo basadas en el principio del argumento son aplicables, por ser técnicas que dan cuenta del número de singularidades de la función dentro de una curva rectificable observando la evolución del argumento de la función a lo largo de dicha curva.

Así, aplicando el principio del argumento sobre la curva generalmente conocida como camino de Nyquist (una curva que encierra todo el semiplano derecho de la hoja principal de Riemann), se puede determinar la estabilidad del sistema en lazo cerrado sin más que determinar el número de revoluciones de la curva resultante de la evaluación alrededor del punto crítico $(-1, j0)$.

2.5 Análisis de la respuesta

Respuesta en el dominio del tiempo como ya se ha comentado en los apartados anteriores, la forma de la respuesta dependerá de la localización de las raíces de la ecuación característica, pudiéndose presentar los siguientes casos:

- No hay raíces en la hoja principal de Riemann. En este caso, la respuesta será una función monótonamente decreciente.
- Hay raíces en la hoja principal de Riemann, localizadas en $(s) < 0$, $Im\,(s) = 0$. En este caso, la respuesta será una función monótonamente decreciente.
- Hay raíces en la hoja principal de Riemann, localizadas en $(s) < 0$, $Im\,(s) \neq 0$. En este caso, la respuesta será una función con oscilaciones amortiguadas.
- Hay raíces en la hoja principal de Riemann, localizadas en $(s) = 0$, $Im\,(s) \neq 0$. En este caso la respuesta será una función con oscilaciones de amplitud constante.
- Hay raíces en la hoja principal de Riemann, localizadas en $(s) > 0$, $Im\,(s) \neq 0$. En este caso la respuesta será una función con oscilaciones de amplitud creciente.
- Hay raíces en la hoja principal de Riemann, localizadas en $(s) > 0$, $Im\,(s) = 0$. En este caso la respuesta será una función monótonamente creciente.

Para el caso particular de los sistemas de orden conmensurable, la respuesta impulsiva podrá expresarse como:

$$g(t) = \sum_{k=0}^{n} r_k\, t^{a-1} E_{a,a}\,(\lambda_k t^a)$$

Siendo λ_k los polos de la ecuación característica, r_k los residuos que resultan de la descomposición en fracciones simples de dicha expresión, y $E_{a,a}(.)$ la función de Mittag- Leffler de dos parámetros definida como:

$$E_{\alpha,\beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{r(ak+\beta)}, \alpha > 0, \beta > 0$$

La forma de estas respuestas será:

1) Monótona decreciente si $|\arg(\lambda_k)| \geq \alpha\pi$.

2) Oscilatoria de amplitud decreciente si $a\frac{\pi}{2} < |\arg(\lambda_k)| < a\pi$

3) Oscilatoria de amplitud constante si $|\arg(\lambda_k)| < a\frac{\pi}{2}$

4) Oscilatoria de amplitud creciente si $|\arg(\lambda_k)| < a\frac{\pi}{2}$, $|\arg(\lambda_k)| \neq 0$

5) Monótona creciente si $|\arg(\lambda_k)| = 0$

La respuesta al escalón responde a la expresión:

$$y(t) = \sum_{k=0}^{n} r_k t^a E_{a,a+1}(\lambda_k t^a)$$

Por otra parte, es fácil entender que la técnica del lugar de las raíces se puede aplicar a los sistemas de orden conmensurable con la misma facilidad que a los sistemas de orden entero.

Lo único que cambia es la interpretación, es decir, la relación de los puntos del plano complejo $\lambda = s^a$ con las características dinámicas del sistema.

CAPÍTULO III. CÁLCULO FRACCIONARIO

3.1 Introducción

Para modificar adecuadamente la dinámica de un proceso y que las salidas sean las deseadas se precisan fundamentalmente dos cosas:

- Una referencia de funcionamiento, modelada por medio de las especificaciones.
- Un controlador que haga que ese objetivo se consiga.

El control fraccionario propone el uso de operadores y sistemas fraccionarios en ambos cometidos, es decir, como sistemas de referencia y como controladores. Como ejemplo de ello y por tratar en este trabajo de tesis sólo los aspectos fundamentales, en lo que sigue se hablará principalmente del integrador fraccionario como sistema de referencia, y de los controladores Proporcionales, Integrales y Derivativos Fraccionarios (FOPID), por incluir este controlador de tres términos todas las acciones básicas de control.

3.2 El integrador fraccionario como sistema de referencia

Con base en lo planteado se ha podido comprobar que el sistema con función de transferencia:

$$G(s) = \frac{A}{s^a + A}, 0 < a < 2$$

que corresponde a la ecuación bajo condiciones iniciales nulas, puede exhibir dinámicas que van desde la relajación hasta la oscilación, incluyendo las dinámicas correspondientes a los sistemas de primer orden y segundo orden como casos particulares. Resulta por tanto interesante tomar este sistema como sistema de referencia. Constituye el punto de partida

para el control, así como un criterio para sintonizar controladores PID tradicionales. Tal función se puede considerar como el resultado de conectar en lazo cerrado un integrador fraccionario de ganancia y orden o, es decir, un sistema cuya función de transferencia tiene la forma:

$$F(s) = \frac{A}{s^a}, 0 < a < 2$$

La función de transferencia (s) es una función de transferencia ideal u óptima, cuyas características principales son un margen de fase constante, $\phi m = \pi(1 - \frac{a}{2})$ que depende sólo del orden de integración α, y una frecuencia de cruce por cero decibelios que depende de la ganancia A.

En el dominio del tiempo esto equivale a una respuesta escalón de la forma:

$$y(t) = Ar_k t^a E_{a,a+1}(-At^a)$$

Que tiene una sobre oscilación o máximo que depende sólo de α, siendo A el factor que modula la rapidez.

3.3 Controladores fraccionarios

Las primeras aportaciones en este campo fueron los artículos que a finales de los años cincuenta y principios de los sesenta que fueron publicados por Tustin, Carlson y Halijak. El primero de ellos, al tratar del control de posición de un servomecanismo hablaba de la conveniencia de un controlador que proporcionase un margen de fase constante sobre un intervalo de frecuencias, lo que equivale a tomar como referencia para el sistema controlado una aproximación del integrador fraccionario comentado en la sección anterior. En los otros trabajos se estudiaba la aplicación del integrador fraccionario para el control de un servomecanismo, y el uso de controladores fraccionarios en sistemas con saturación. Tras estas aportaciones pioneras y aisladas hubo que esperar una década para que aparecieran las primeras aportaciones sistemáticas del uso de controladores fraccionarlos en los trabajos de Oustaloup y sus colaboradores. A partir de los años ochenta han ido apareciendo algunos otros trabajos que consideran la posibilidad de utilizar controladores fraccionarios, utilizando métodos y técnicas que van desde el lugar de las raíces hasta la optimización H_∞.

3.3.1 Acciones básicas de control

Partiendo del diagrama de bloques de la Figura (anterior), se establecen aquí los efectos de las acciones básicas de control del tipo K^{s^μ} para $\mu \in [-1,1]$. Las acciones básicas de control tradicionalmente consideradas serán casos particulares de este caso general, en los que para:

μ = 0 se ejercerá una acción proporcional, para

μ = -1 se ejercerá una acción integral, y para

μ = l una acción derivativa.

Como es sabido, los principales efectos de la acción integral son hacer más lenta la respuesta del sistema, disminuir la estabilidad relativa del mismo, y eliminar el error estacionario del sistema ante entradas para las que antes tenía un error finito. Dichos efectos se pueden ver en los distintos dominios de análisis. En el dominio del tiempo, los efectos sobre la respuesta transitoria se manifiestan en un decrecimiento del tiempo de subida y un aumento del tiempo de establecimiento y la sobreoscilación. En el plano complejo, los efectos de la acción integral se traducen en un desplazamiento del lugar de las raíces del sistema hacia el semiplano derecho. Por último, en el dominio de la frecuencia dichos efectos se manifiestan en un incremento de $-20dB/dec$ en las pendientes de la curva de magnitud y un descenso de $\pi/2$ en la curva de fase. En el caso de que se tenga un orden de integración fraccionario, es decir, $\mu \in [-1,0]$, la selección del valor de μ se traduce en una determinada ponderación de los efectos arriba comentados.

En el dominio del tiempo, los efectos de la acción de control pueden inducirse de la manera en que dicha acción trata a una señal error que es una onda cuadrada. Como puede observarse, los efectos de la acción de control sobre la señal error asumida, varían entre los efectos de una acción proporcional ($\mu = 0$) y los de una acción integral ($\mu = -1$), curva a tramos rectos), mientras que para valores intermedios de y la acción de control es creciente para un error constante, lo que resulta en la eliminación del error estacionario, y decrece cuando el error se hace cero, lo que resulta en una menor inestabilidad del sistema.

En el plano complejo, se puede ver que la selección de un valor de μ se traduce en la elección de en qué medida se desplaza el lugar de las raíces hacia el semiplano derecho, y de los valores de K que hacen que se cumpla la condición de módulo.

3.3.2 Controladores PID fraccionarios

El controlador PID, que engloba las tres acciones básicas de control, es todavía el más utilizado en la industria de control de procesos debido, fundamental- mente, a su simplicidad funcional y a la robustez que proporciona a los sistemas de control, y ha recibido una atención constante desde que Ziegler y Nichols presentaron sus métodos de sintonía en 1942. Utilizando los operadores fraccionarios, es posible obtener una estructura más general para el PID clásico que, manteniendo la simplicidad del concepto y lo compacto de su formulación, hace posible tratar con una clase más general de problemas de control en los cuales la naturaleza fraccionaria del controlador puede venir impuesta, bien por la naturaleza fraccionaria del propio sistema a controlar o bien por la especial naturaleza de las respuestas temporales o frecuenciales requeridas.

El controlador PID clásico puede considerarse como una forma particular de compensación atraso-adelanto en el dominio de la frecuencia. Su función de transferencia puede expresarse como:

$$C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + k_{d^s}$$

O bien:

$$C(s) = k\frac{(\frac{s}{wc})^2 + \frac{2d_{c^2}}{wc} + 1}{s}$$

Con $w_c = \sqrt{k_i/_{k_d}}, d_c = k_p/2\sqrt{k_1}k_{d,}k = k_i$

Por tanto, las aportaciones del controlador pueden verse como dependientes de:

a) las ganancias $\boldsymbol{k_{p,}k_{i,}k_d}$

b) la ganancia k y los parámetros $\boldsymbol{W_{c,}} \boldsymbol{d_{c,}}$

En la respuesta en frecuencia del controlador, la elección de estas ganancias o parámetros es equivalente a la selección de la posición, suavidad y valor del mínimo en la curva de magnitud, y la pendiente de la curva de fase a la frecuencia en que se produce tal mínimo. Sin embargo, para altas y bajas frecuencias, los valores de las pendientes en la curva de magnitud y de las aportaciones de fase son fijos.

La ecuación íntegro-diferencial que define la acción de control del FOPID es:

$$u(t) = K_p e(t) + K_i D^{-\lambda} e(t) + K_d D^{\mu} e(t)$$

Aplicando la transformada de Laplace a la ecuación anterior con condiciones iniciales nulas, se obtiene la función de transferencia del controlador, que puede expresarse de la forma:

$$C_f(s) = K_p + \frac{K_i}{s^{\lambda}} + K_d s^{\mu}$$

$$= k \frac{(\frac{s}{w_f})^{\lambda+\mu} + 2d_f s^{\lambda} + 1}{s^{\lambda}}$$

Como puede observarse, el uso del FOPID permite seleccionar, además de lo que permitía el PID clásico, bien las pendientes de la curva magnitud, bien las aportaciones de fase a bajas y altas frecuencias. De una forma gráfica, se pueden ver las posibilidades ofrecidas por el FOPID en la Figura 8. En ella, estas nuevas posibilidades se presentan gráficamente como la posibilidad de ocupar toda la superfice del cuadrado definido por los vértices que representan las únicas posibilidades del PID clásico.

Desde los trabajos de Ziegler y Nichols, se han propuesto muchos métodos de diseño y procedimientos de sintonía para los controladores PID. Teniendo en cuenta la posibilidad de seleccionar no sólo las constantes del controlador, sino también los órdenes de las acciones integral y derivativa, se pueden generalizar estos métodos y procedimientos de forma que puedan hacerse más flexibles y potentes, y permitan resolver una clase más amplia de problemas de control. A modo de ilustración, podemos referirnos que resolviendo un problema de optimización no lineal, se pueden sintonizar los parámetros de un FOPID para que el sistema controlado cumpla especificaciones de diseño, incluyendo entre ellas especificaciones de error estacionario, márgenes de fase y ganancia, frecuencias de cruce, fase plana alrededor de la frecuencia de cruce de ganancia, atenuación de ruido de alta frecuencia, limitación de los efectos de las perturbaciones, y limitación del esfuerzo de control. En la literatura técnica se pueden encontrar otros criterios y métodos para sintonizar controladores FOPID.

Otras estrategias de control fraccionario Teniendo en cuenta la forma de las acciones básicas de control, es fácil entender que el uso de los operadores fraccionarios se extienda a otras estrategias de control diferentes al control PID, al menos a todas aquellas en las que se haga uso de una derivada o una integral. De entre todas las que se han propuesto, es de especial relevancia el control CRONE (Commande Robuste dÓrdre Non Entier). Hasta el día de hoy existen tres generaciones del control CRONE. La primera generación considera el empleo de un controlador del tipo:

$$C(s) = As^{a}$$

Siendo su propósito aumentar el margen de fase del sistema en un cierto valor determinado por α. Este tipo de controladores resultan muy útiles cuando la planta a controlar tiene una fase constante, al menos en un rango de frecuencias alrededor de la frecuencia de cruce por 0dB, en

cuyo caso el sistema será robusto ante cambios en la ganancia de la planta. Cuando la planta a controlar, G(s), no presenta fase plana alrededor de la frecuencia de interés, el controlador CRONE de primera generación no puede asegurar la robustez ante cambios en la ganancia del sistema. El control CRONE de segunda generación se basa en la identificación frecuencial del controlador C(s) que hace posible que el sistema en lazo abierto sea del tipo $F(s) = C(s)G(s) = As^a$, al menos en el rango de frecuencias apropiado. Tanto la primera como la segunda generación CRONE tienen por propósito que el sistema controlado sea robusto ante cambios en la ganancia de la planta.

Sin embargo, hay otros tipos de incertidumbres en el modelo que no se cubren con estas dos estrategias de control. La tercera generación CRONE se centra en este aspecto, considerando incluso órdenes de derivación e integración complejos (ver

Otra de las estrategias en la que el cálculo fraccionario ha encontrado aplicación es el control adaptativo. Como ya se conoce, este tipo de control se basa, de manera general, en expresar el comportamiento deseado del sistema en términos de un modelo de referencia que describe las propiedades entrada-salida del sistema en lazo cerrado. Los parámetros del controlador se ajustan de manera que el error entre las salidas del sistema real y el de referencia sea mínimo. En algunos trabajos se propone el empleo del cálculo fraccionario tanto para el ajuste de los parámetros del controlador como para la definición de los sistemas de referencia.

El control iterativo también ha sido objeto de estudio para el caso de sistemas de orden no entero. En (Chen and Moore, 2001) se propone emplear un derivador fraccionario para este tipo de control, y en (Lazarevit, 2004) se presenta un nuevo algoritmo para el control por aprendizaje teniendo en cuenta dicho derivador de orden no entero.

El estudio de compensadores en adelanto y atraso generalizados de orden fraccionario es otra de las aportaciones a destacar. Su estructura y características frecuenciales, así como técnicas para la sintonía de los mismos se describen. Destacándose la mayor robustez que ofrece este tipo de compensadores frente a los de orden entero.

También cabe mencionarse la introducción del cálculo fraccionario en control de sistemas no lineales, destacándose las aportaciones en lo que se refiere al control en modo deslizante para el trabajo el uso de operadores fraccionarios tanto para determinar la superficie de deslizamiento como para la ley de control.

El observador de perturbación es otra de las estrategias consideradas. En este tipo de control, se trata la diferencia entre la salida actual del sistema y la salida del modelo nominal como una perturbación equivalente que se aplica a dicho modelo nominal. Una vez estimada la perturbación se emplea dicha estimación como una señal de compensación.

CAPÍTULO IV. ESPECIFICACIONES DE UN SEGUIDOR DE LINEA

Un seguidor de línea es un robot cuya finalidad es la de seguir una línea dibujada o trazada en el piso, esta línea servirá como referencia para que el robot transite por un área designada, pudiendo moverse por líneas tanto rectas como curvas teniendo como apoyo sensores para detectar la línea trazada y realizar su desplazamiento por ella, sin tomar en cuenta el color del fondo por el que transite.

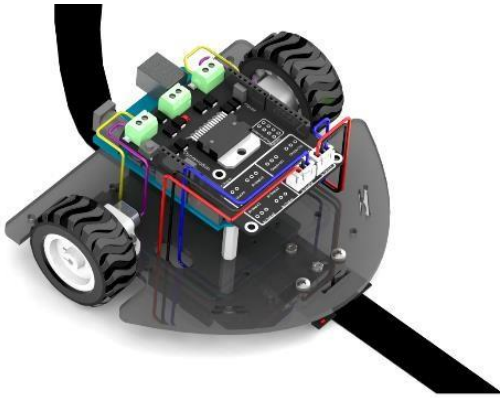


Imagen 1. Seguidor de Línea.

4.1 ¿En qué sectores se utilizan?

El área de trabajo para el campo de la robótica es muy extenso, y el caso del seguidor de línea no es ajeno a esto, por tanto a continuación se mencionan algunos campos de aplicación para el seguidor de línea, no sin antes mencionar que pueden ser aún más, todo en base al ingenio que se aplique para un mayor desarrollo de este robot.

- Educación.
- Industria.
- Entretenimiento.

4.1.1 ¿Cómo es el funcionamiento?

El funcionamiento es realizado por medio de dos motores y una línea de sensores los cuales son los encargados de realizar el movimiento y recopilar información del trayecto, para realizar estas funciones se debe tener en cuenta que el camino puede ser trazado de dos formas, un fondo blanco con una línea negra o un fondo negro con una línea blanca. Al colocar el seguidor de línea sobre estas superficies a las cuales se denominaran como "pistas" la barra de sensores deberá realizar por medio del operador o usuario una calibración al fondo, una a la línea y la activación de la turbina, para que el microcontrolador recolecte esta información y la procese a modo de que el seguidor de línea no salga en ningún momento de la pista realizando su recorrido en el menor tiempo posible.

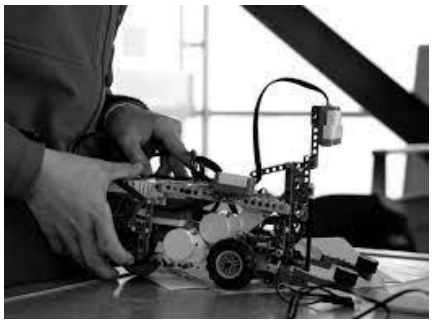


Imagen 2. Usos del Seguidor de Línea

4.1.2 Elementos a considerar en un seguidor de línea

Los factores más importantes a considerar para un seguidor de línea son el diseño electrónico y mecánico, la importancia del diseño es fundamental para la ubicación de los sensores pues son aquellos los encargados de ubicar de forma correcta la línea por la cual se desplazara el seguidor de línea. Se deberá utilizar programas que permitan el diseño de la placa, editar sus componentes, unirlos e incluso tener modelos en 3D. Con un diseño adecuado el seguir de línea podrá censar de forma correcta su trayectoria y corregir de forma eficiente cuando salda de la línea. El diseño

también influye en la aerodinámica del seguir, la selección de componentes y la distribución del peso por la placa también son factores. Se necesitara un microcontrolador que cumpla la función de cerebro, recopile la información y distribuya las indicaciones a todos los componentes del seguidor para que realicen su trabajo. Cuando el seguidor salga de su trayectoria inmediatamente se enviara a los motores la orden de reducir, mantener o aumentar la velocidad dependiendo del caso y es el microcontrolador el encargado.

4.1.3 Presupuesto de componentes para seguidor de línea velocista

Presupuesto Seguidor de línea Velocista			
Componentes	**Mercado Libre**	**Amazon**	**Otros**
Arduino Nano	**$249**	**$254**	
Puente H Driver TB6612FNG	**$115**	**$178.5**	
Micromotores Pololu 10:1 HPCB 6V (2)	**$449**	**$245 c/u**	
Llantas de caucho silicón (2)	**$390**	**----------**	
Brackets para micromotores pololu	**Incluidos con los motores**	**$219**	
Bateria Li-Po 2S 7.4 V 300 mAh	**$399**	**----------**	

Control y arrancador	**----------**	**----------**	
Soportes para barras de sensores	**$85**	**----------**	
Turbina EDF30	**----------**	**----------**	
Juego de convertidores de conector FCC/FCC 0.5 mm a 2.54 mm	**$175**	**----------**	
Resistor de 10 Kohm	**$0.50 c/u**	**$0.50**	
Push Boton 2 pines	**$5**	**$5**	
Switch de 3 pines	**$8**	**$8**	
Barra de 16 sensores con nux	**$741**	**----------**	
ESC			

Tabla 1. Lista de componentes para el Proyecto.

La tabla anterior es un presupuesto que muestra una lista de todos los componentes que se utilizan en un Seguidor de Línea Velocista, además de los precios de cada componente encontrados en páginas de tiendas por internet y un distribuidor extra para conocer en primera instancia los elementos a utilizar para el seguidor, y también tener un presupuesto estimado del precio que implica realizar el seguidor, no obstante cabe recalcar que estos precios están sujetos a cambios y que los componentes pueden o no estar disponibles, también se pueden comprar en tiendas de electrónica o con distribuidores.

4.2 Creación de componentes en proteus (pcb layout)

Para desarrollar el Seguidor de Línea Velocista será necesario el diseño de una placa PCB en donde se montarán todos los componentes a utilizar. Para realizar esta tarea se utilizará el software de electrónica Proteus. Con este programa lo primero es desarrollar los componentes de la tabla de presupuesto debido a que no se encuentran en el software, y son necesarios para el desarrollo del Seguidor. Lo que se puede observar en la imagen 3 es el diseño de la placa terminada (chasis) del seguidor de línea, junto con los componentes utilizados entre los cuales podemos observar el Arduino Nano, el puente H (Driver) los motores, las llantas, la turbina, etc.

El diseño de los componentes es la base para el diseño de la placa o chasis, porque en base a ello se podrá definir la ubicación de cada uno de los componentes así como también se definen cada una de las perforaciones que debe llevar, con el fin de cumplir con las especificaciones requeridas para los Seguidores de Línea Velocistas tanto para concurso como las especificaciones necesarias para los fabricantes de PCB

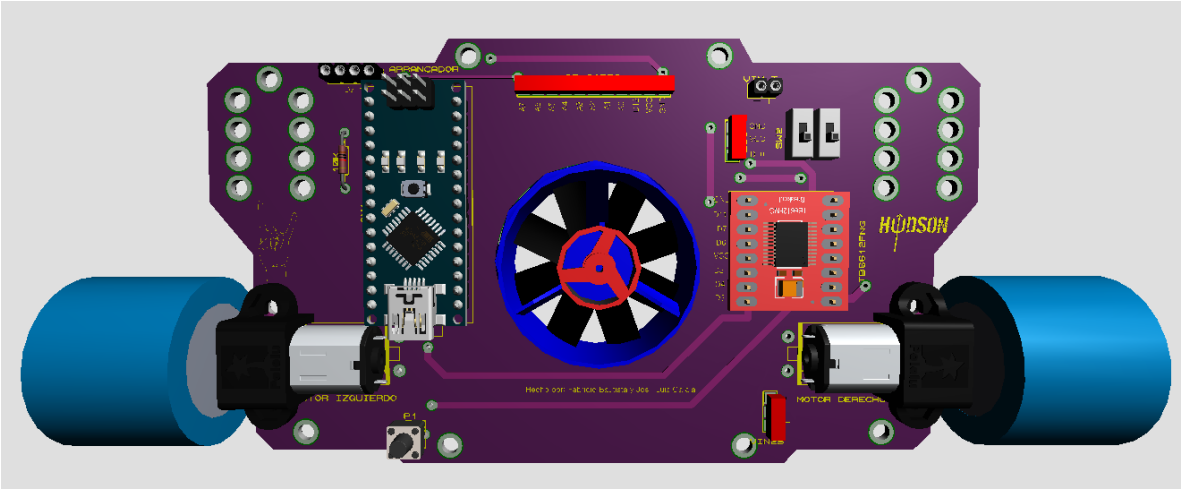


Imagen 3. Modelado 3D de placa para seguidor de línea velocista.

4.2.1 Empaquetados

Se llamara empaquetado a cada uno de los componentes electrónicos utilizados en el Seguidor de Línea, desde el más simple hasta el más complejo, ya sea el Arduino Nano o la turbina, un Switch o el pulsador, además de que dichos empaquetados deberán ser realizados en Proteus debido a que el programa no cuenta con ellos por defecto. Como muestra de empaquetados se puede tomar como referencia la imagen 3 del modelado 3D, donde todos los componentes sobre la placa morada son empaquetados, y la imagen 4 siguiente, donde se observan las perforaciones correspondientes a los empaquetados sobre la placa y en líneas de color amarrillo se tienen especificados los espacios de destino para cada uno de ellos.

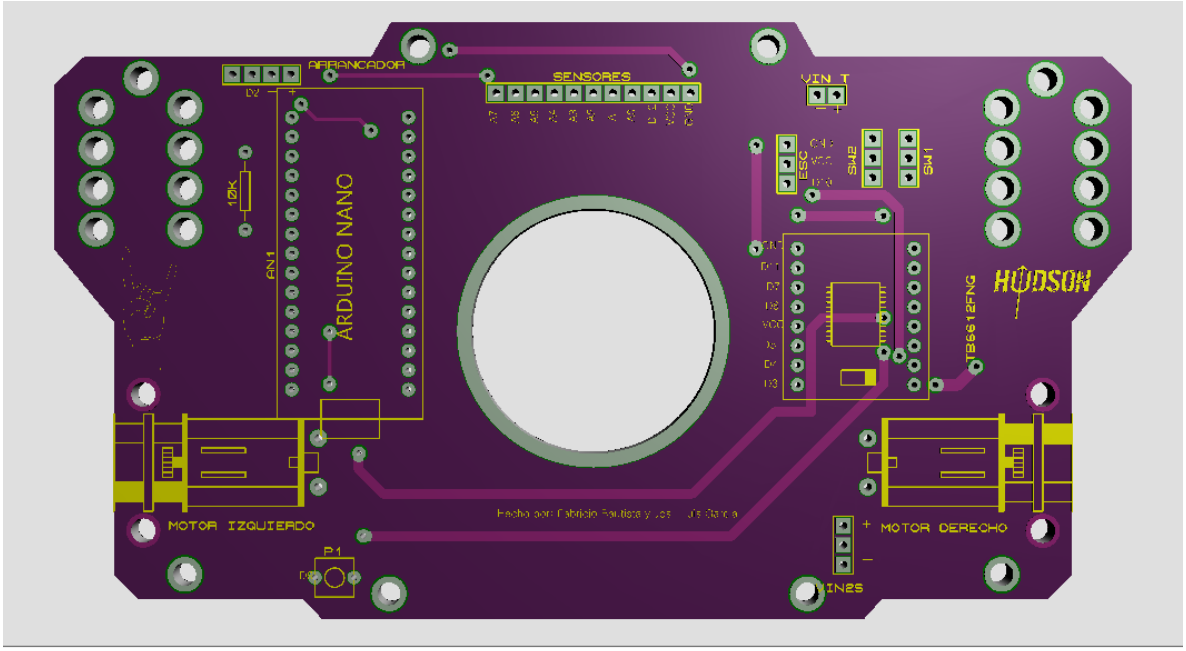


Imagen 4. Empaquetado de placa para seguidor de línea velocista.

Para realizar los empaquetados se utilizará la sección Ares de Proteus o también conocida como PCB Layout que se encuentra ubicada en la parte superior del programa justo a un costado de la

opción para la página de inicio o Home, guiándonos por un icono en color rojo que emula a un componente electrónico señalado por un recuadro rojo.

Imagen 5. Barra de opciones Proteus.

4.2.2 Archivos .STEP

Para iniciar a realizar el proceso de los empaquetados serán necesarios algunos archivos .STEP que son con los que trabaja el programa, además de unas imágenes de los componentes a realizar que se deben insertar o anexar dentro de la carpeta de origen del mismo.

Los siguientes son los pasos a realizar para insertar estos archivos.

1. Los archivos requeridos deben descargados de internet y ubicados en una carpeta específica, su formato es .STEP para ser reconocidos por el programa, y en .JPG o en su defecto .PNG para las imágenes de los empaquetados, se debe seleccionar y copiar o cortar todos los archivos de esta carpeta para pegarlos en la carpeta destino del programa, tal como se muestra en la siguiente imagen:

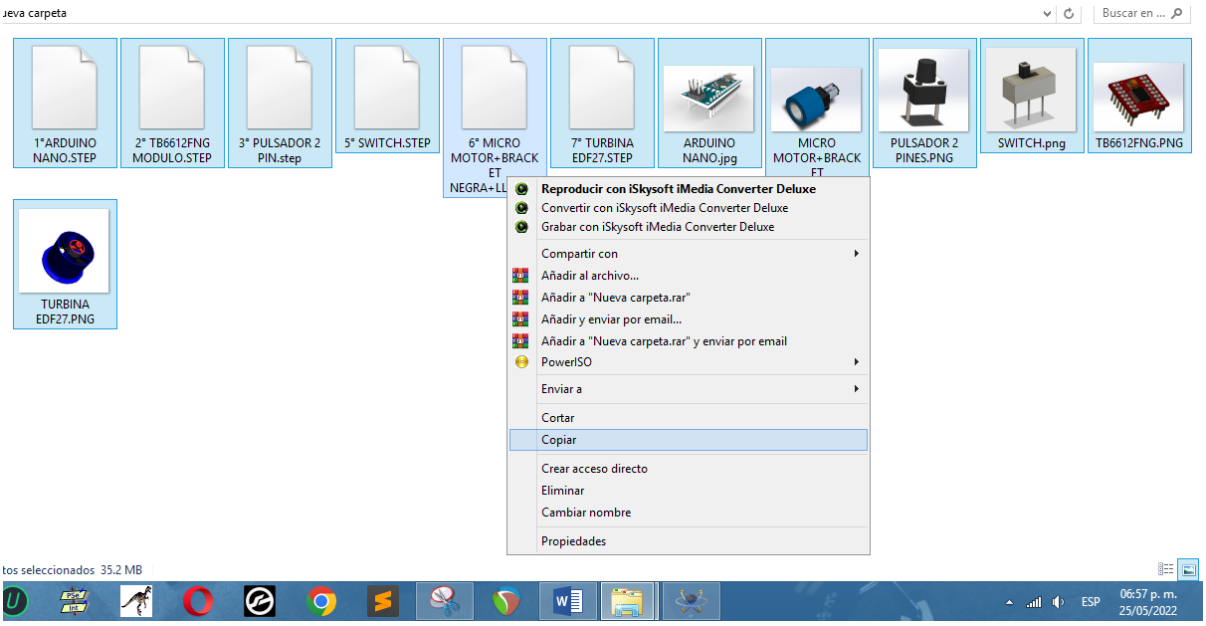


Imagen 6. Archivos .STEP, .PNG y .JPG.

2. Una vez seleccionada la opción para copiar/cortar se debe ubicar el apartado “Equipo” y seleccionar la opción “Windows (C:)” ahí dentro se desplegaran muchas carpetas siendo un equipo de 64 bits se debe abrir la carpeta “Program Files (x86)” e ingresar a ella.

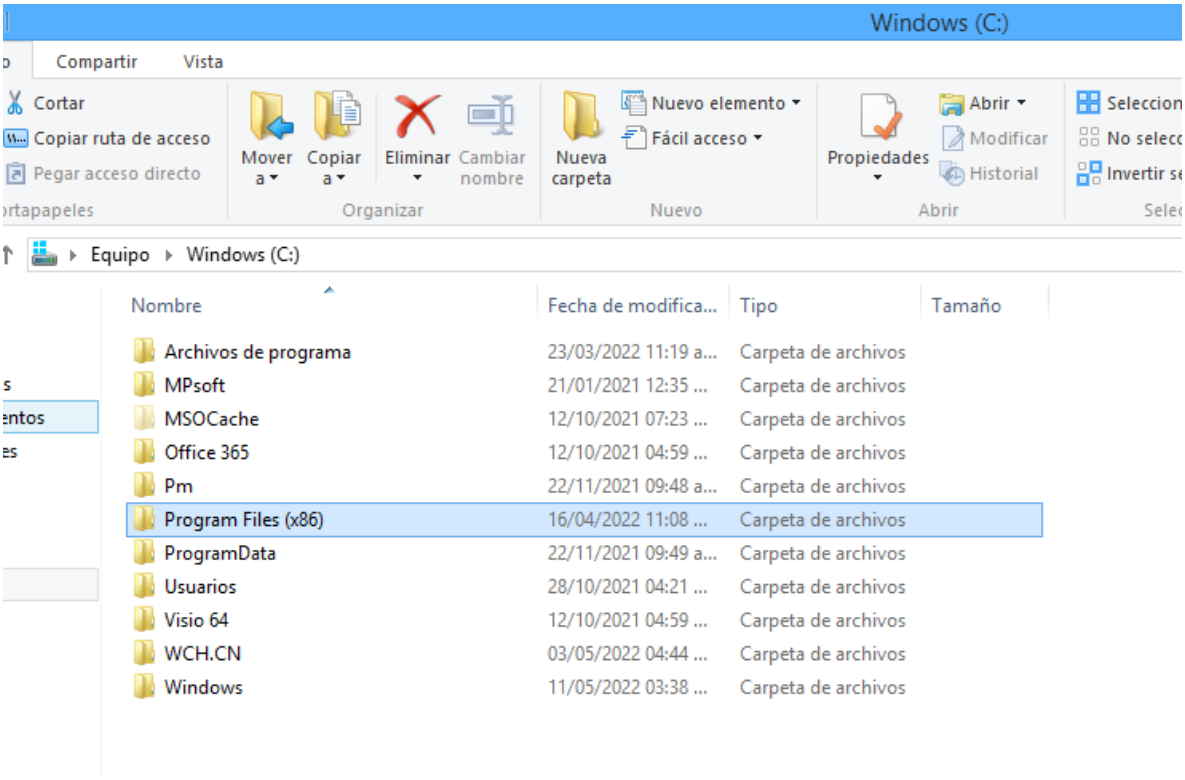


Imagen 7. Carpeta Program Files (x86).

3. Posteriormente dentro de la carpeta “Program Files (x86) se tiene que accesar a las carpetas con el siguiente orden: “Lab Center electronic” > Proteus 8 Professional > DATA >MCAD. Una vez dentro de la carpeta de MCAD lo único restante por realizar es pegar los archivos .STEP, .JPG y .PNG de la primer carpeta de donde se copiaron cortaron, quedando de la siguiente manera:

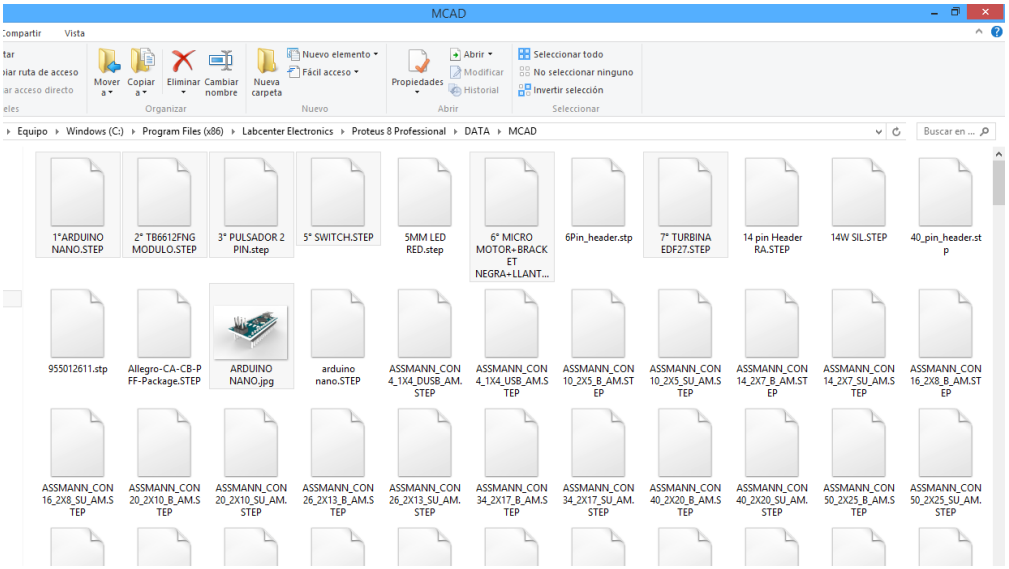


Imagen 8. Programas anexados a Proteus.

4.2.3 Arduino Nano

Una vez realizados los pasos anteriores lo siguiente será volver a Proteus e ir a la extensión PCB Layout. Un dato muy importante son las dimensiones de cada uno de los componentes, tomando como referencia para el Arduino Nano una imagen de internet con las dimensiones respectivas a este componente electrónico, se puede partir para el desarrollo de su respectivo empaquetado.

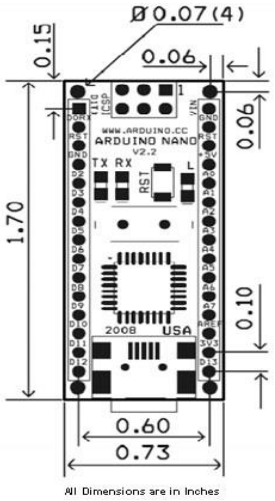


Imagen 9. Dimensiones del Arduino Nano.

En la imagen se observan las dimensiones del Arduino Nano dadas en milipulgadas, medida con la que se trabajara para los empaquetados en Proteus seleccionando la opción del sistema Inglés para trabajar acorde a las dimensiones presentadas en la imagen, sin crear conflicto por medidas diferentes. Esta opción se encuentra ubicada en la parte superior de la pantalla, señalada por una "m" dando la opción a elegir entre Métrica e Imperial.

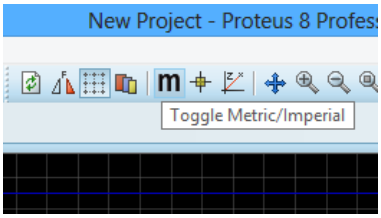


Imagen 10. PCB Layout y sistema inglés.

Para iniciar con el empaquetado del Arduino Nano se ubicara al apartado de opciones que se encuentra en la parte izquierda de programa, justo en la opción "Round Through Pad Mode" y se seleccionara la opción 70-30 para el diámetro que corresponde a los Pads de perforación para el Arduino Nano. .

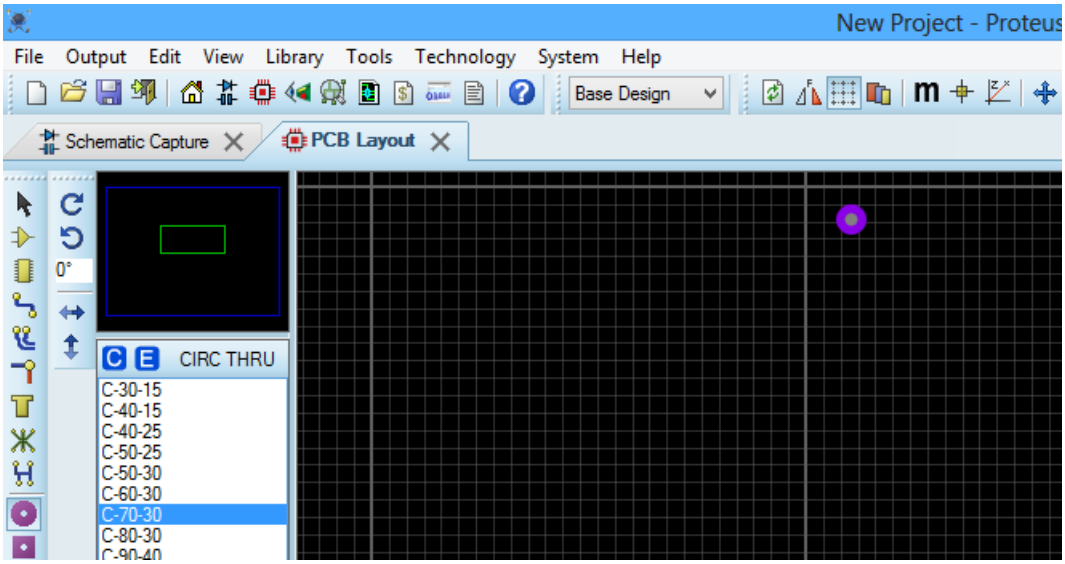


Imagen 11. Pad Mode 70-30.

Se deben colocar 15 puntos en forma horizontal para el empaquetado con una distancia entre Pads de perforación de 100 milipulgadas, para que sea más fácil colocar estos puntos se puede cambiar el tamaño de la cuadricula en View > Snap to 50 th para tener una distancia de 50 mili pulgadas. Se deben seleccionar todos los Pads de perforación para duplicarlos y colocarlos en la parte de abajo de la primera fila de Pads colocados, debido a que el Arduino lleva esas perforaciones por ambos lados, posteriormente se procede a dar la distancia correspondiente entre Pads, teniendo una distancia de 600 milipulgadas entre ellos.

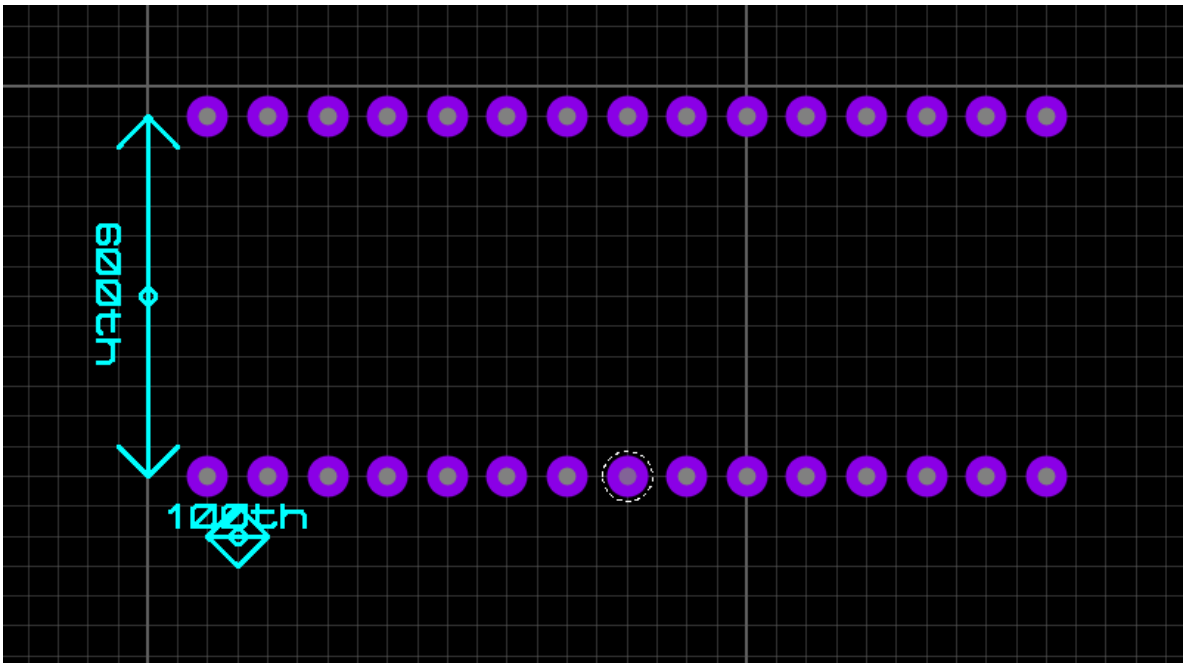


Imagen 12. Distancia de los puntos.

Con la herramienta de distancias se puede verificar que tanto los Pads como la fila de Pads tienen las distancias correctas. Posteriormente con la herramienta generadora de nombres se enumeran cada uno de los puntos comenzando de derecha a izquierda en forma de U, partiendo del número 0.

Se debe colocar un recuadro que contenga a todos los puntos dentro de él y uno más para tener la orientación del Arduino (El recuadro mayor debe ser en base a las medidas del Arduino).

Incluso se puede colocar el nombre del componente como se muestra en la imagen siguiente:

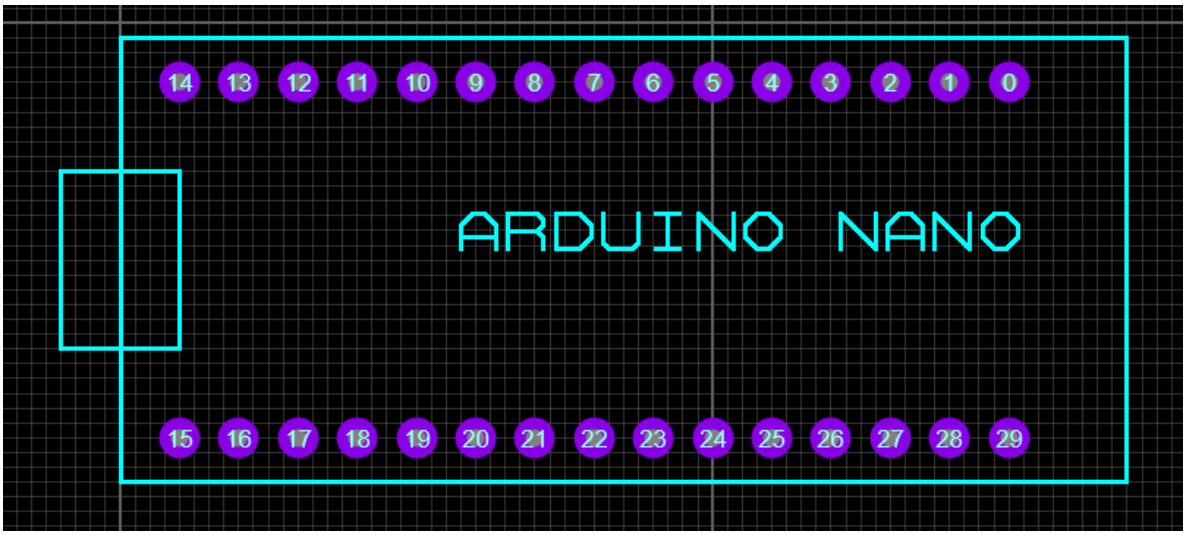


Imagen 13. Modelo del empaquetado.

Para la construcción del empaquetado se selecciona con el click izquierdo del mouse el diagrama del Arduino Nano de forma completa, y posteriormente con click derecho se selecciona la opción “Make Package”, posteriormente saltara un recuadro que se debe llenar con los datos que solicita (se puede tomar de referencia la siguiente imagen). La parte importante es en el apartado Package Type donde tiene que seleccionar la opción Through Hole debido a que se están creando componentes para perforación.

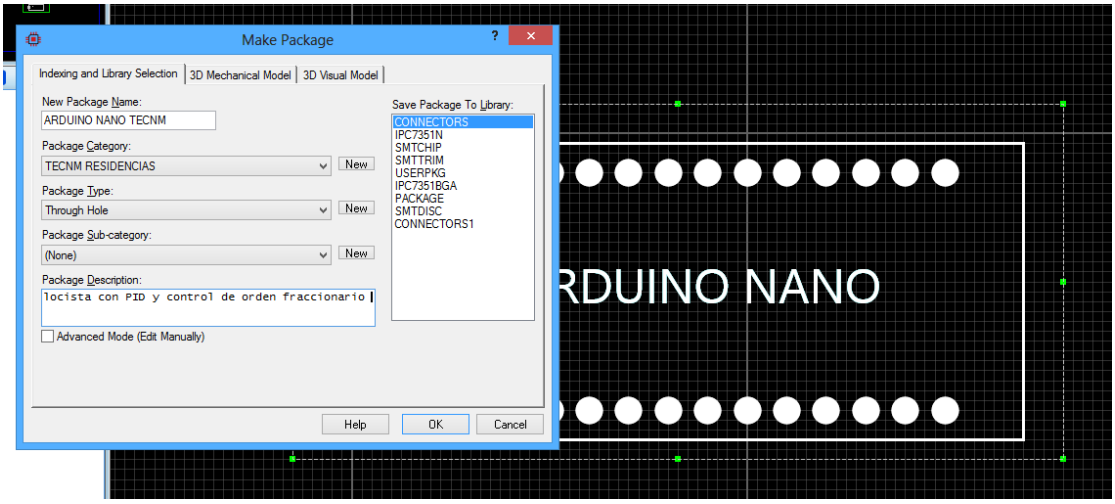


Imagen 14. Construcción del empaquetado.

En la opción superior se encuentra el apartado “3D Visual Mode” se debe dar click en la opción “M-CAD (STEP or IGES) File, click en la figura de carpeta para que abra una ventana a la carpeta M-CAD de las librerías de Proteus ahí se debe seleccionar la imagen del Arduino Nano que previamente se ha colocado tal como lo muestra la siguiente imagen:

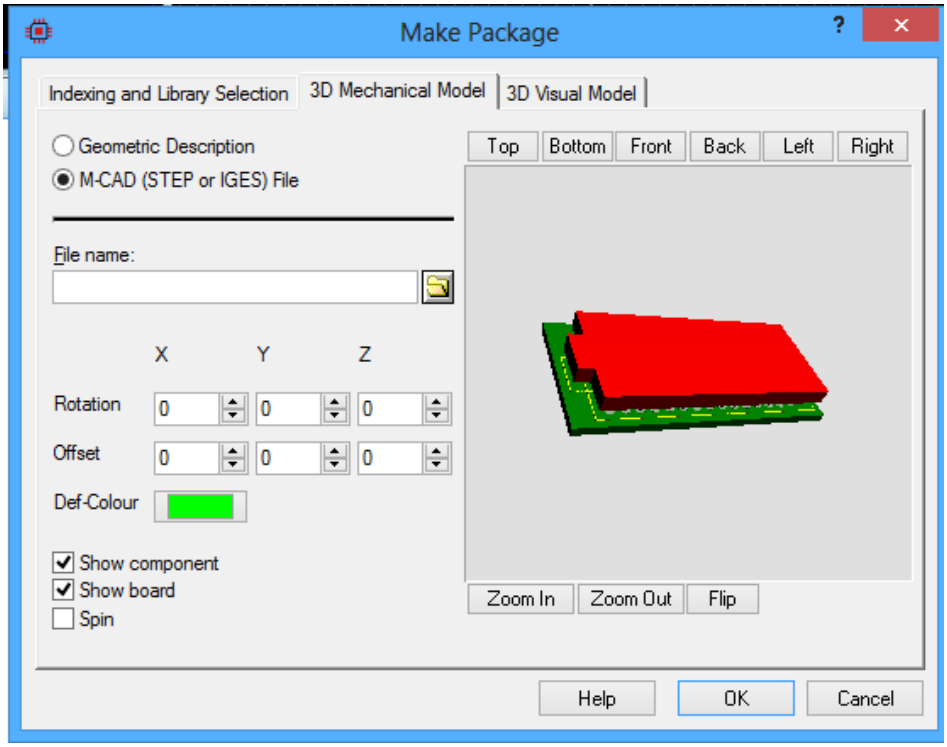


Imagen 15. 3D Visual Mode.

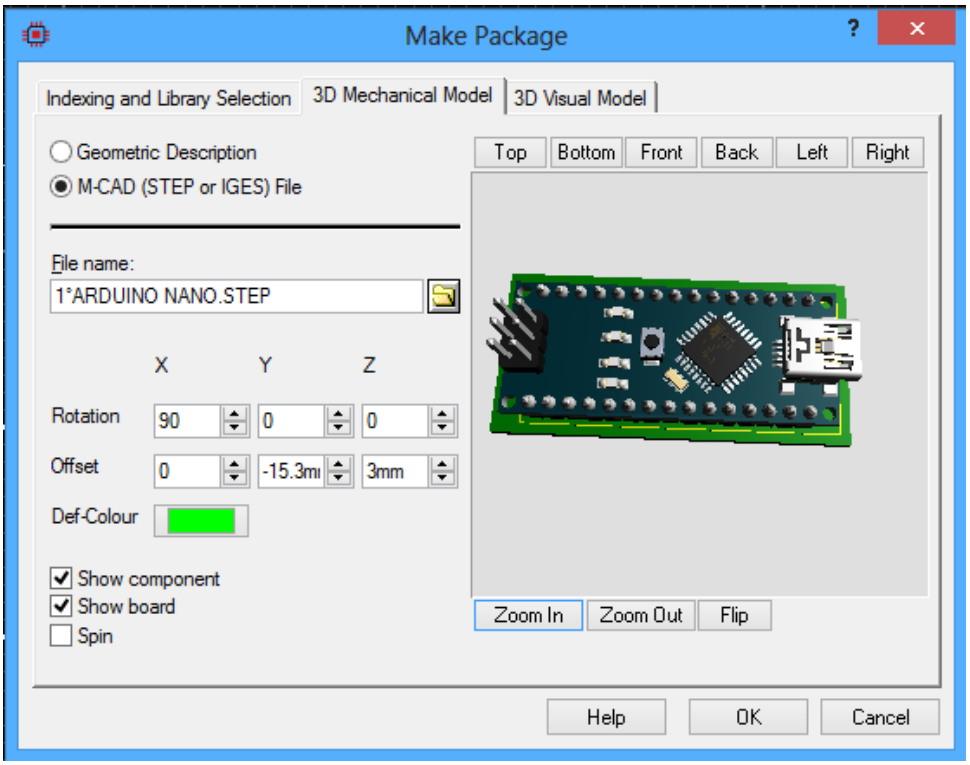


Imagen 16. Imagen cargada.

Es muy importante entender que la imagen del empaquetado debe quedar cuadrada lo más preciso posible a las perforaciones de la placa verde que se muestra por ejemplo, desplazándola sobre 3 ejes, x, y, z tanto en Rotation como en Offset, moviendo los valores de los ejes el empaquetado también se moverá. Es importante que el Arduino Nano quede en la misma dirección y sentido que las líneas amarillas que están dibujadas en la placa ejemplo debajo de él.

Nota: Hasta no estar completamente posicionado el Arduino Nano de forma correcta, no se debe dar “OK”, porque el componente no estaría completo de la manera en que se necesita y tendría un error.

En la siguiente imagen se puede observar cómo queda el empaquetado en 3D utilizando el visualizador 3D de Proteus.

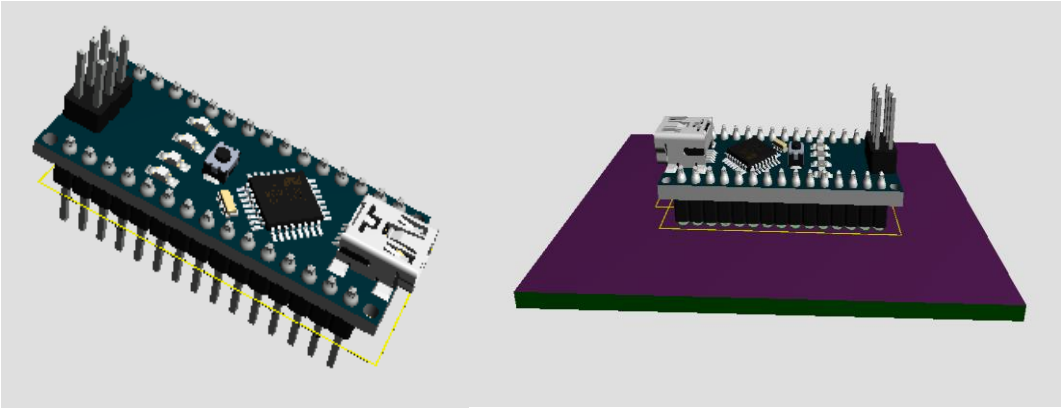


Imagen 17. 3D del empaquetado “Arduino”.

4.2.4 Puente HTB6612FNG

Para la construcción de los empaquetados posteriores ya no será necesario iniciar con la instalación de los archivos, ese proceso solo se realiza una única vez para todos los empaquetados, por lo tanto una vez aclarado este punto, se tendrá que recurrir al conocimiento de las medidas pertinentes del puente TB6612FNG, para esto se puede buscar una imagen en internet con las medidas correspondientes tal como con el Arduino Nano, la imagen es la siguiente:

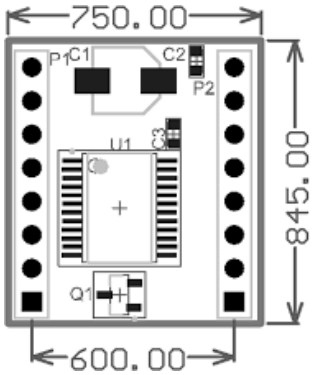


Imagen 18. Puente H TB6612FNG.

En el apartado de PCB Layout de Proteus se deben seleccionar los Pads para perforación utilizados anteriormente, yendo al apartado de opciones que se encuentra en la parte izquierda del

programa, como la opción “Round Through Pad Mode” y se seleccionara 70-30 como diámetro para realizar las primeras perforaciones del puente TB6612FNG.

Se deben colocar 8 puntos en forma horizontal para el empaquetado tal como en el esquema de referencia, con una distancia entre Pads de perforación de 100 milipulgadas. Para que sea más sencillo colocar los Pads de perforación se puede cambiar el tamaño de la cuadricula en View > Snap to 50 th para así tener una distancia de 50 mili pulgadas. Se tiene que duplicar y colocar en la parte de abajo la primera fila de Pads, el puente H lleva estas perforaciones por ambos lados. Posteriormente se procede a darle la distancia correspondiente a los Pads, teniendo una distancia entre ellos de 600 mili pulgadas. Una vez definida la distancia con la herramienta generadora de nombres se enumera cada uno de los puntos comenzando de derecha a izquierda en forma de U iniciando por la fila superior hacia la inferior, partiendo del número 0. Se debe colocar un recuadro que contenga todos los puntos dentro de él y otro para tener la orientación del TB6612FNG, se puede colocar el nombre del componente como se muestra en la imagen siguiente; cabe mencionar que el recuadro mayor debe ser correspondiente a 750 mili pulgadas a lo ancho y 850 milipulgadas a lo largo. En la imagen de medidas del puente H se tiene como referencia 845 milipulgadas pero se pueden redondear a las 850 milipulgadas. Una vez terminado el rectángulo mayor se agregaran un par de rectángulos menores en él, puesto que son componentes del puente H que serán necesarios como referencia de dirección del empaquetado. El esquema no tiene medidas definidas pero se pueden tomar como referencia las siguientes: 325 de largo por 250 milipulgadas de ancho para el primero, y 150 milipulgadas de ancho por 75 milipulgadas de ancho para el segundo. Además, se agregaran unos pines para el rectángulo interior mayor, al igual que con los Pads de perforación los pines se pueden obtener en la barra de herramientas ubicada en la parte inferior izquierda sobre

el icono de una línea que responde a “2D Graphics Line Mode” y se deberán agregar 12 pines por cada lado.

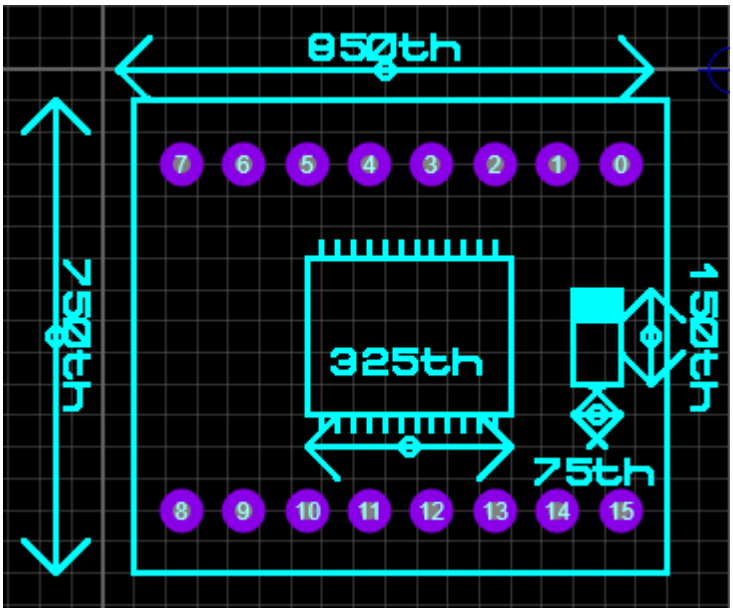


Imagen 19. Medidas del Puente H TB6612FNG.

Una vez colocadas las medidas correspondientes se eliminan para continuar con el proceso del empaquetado, de igual forma que con el Arduino Nano se debe seleccionar por completo el diagrama del puente H aplicar click derecho para utilizar la opción “Make Package” y llenar los campos correspondientes con la información necesaria que pide el empaquetado, se puede tomar como guía o referencia el llenado de datos del Arduino.

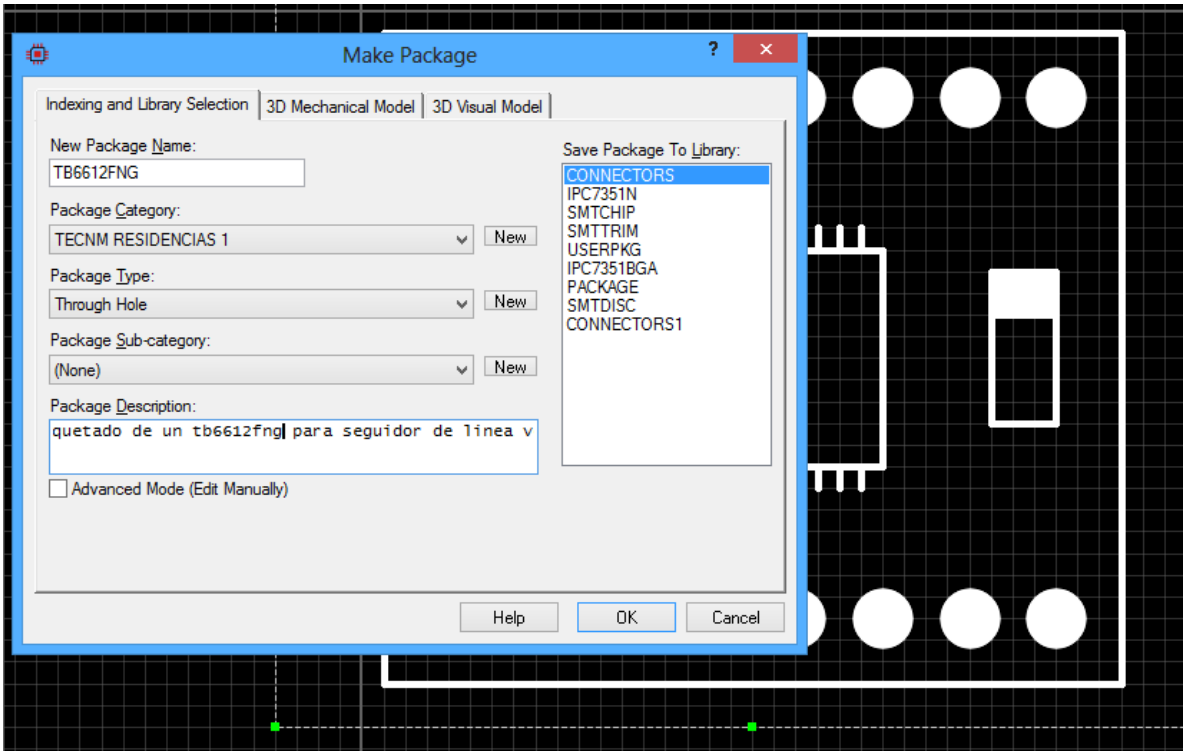


Imagen 20. Empaquetado del Puente H TB6612FNG.

En la opción superior se encuentra el apartado “3D Visual Mode” donde se deberá dar click en la opción “M-CAD (STEP or IGES) File, para después dar click en la figura de carpeta que mostrara una ventana con la carpeta M-CAD de las librerías de Proteus, ahí se selecciona la imagen del puente H que fue previamente colocado.

Nuevamente la imagen debe quedar en la misma dirección a las perforaciones que se muestran en la visualización 3D, para ello se debe desplazar el puente H sobre 3 ejes, X, Y y Z tanto en Rotation como en Offset, incrementando o disminuyendo los valores para que el empaquetado se mueva a la posición correcta. Es importante que el puente H quede en la misma dirección y sentido que las líneas amarillas dibujadas en la parte inferior de la placa.

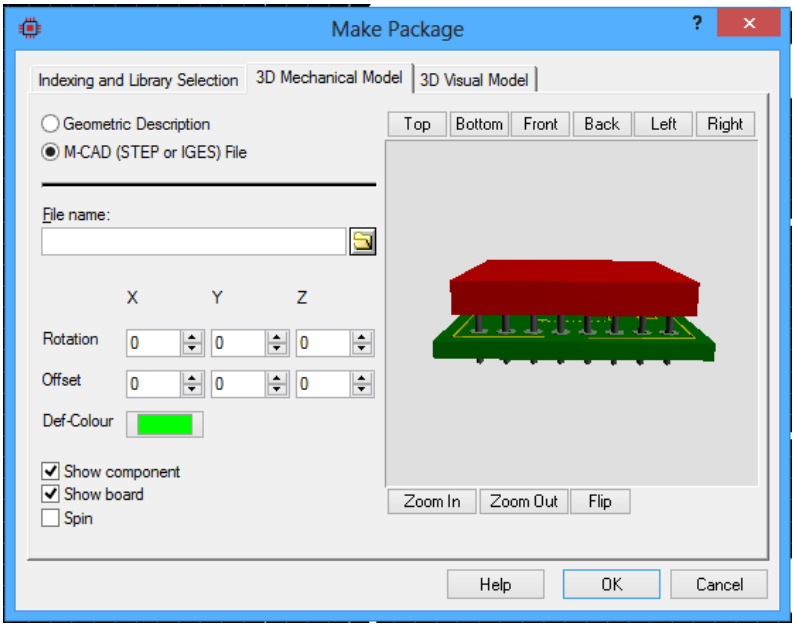


Imagen 21. Visual Mode.

Nota: hasta no estar completamente posicionado el puente H de forma correcta, no se debe dar "OK", porque el componente no estaría completo de la manera en que se necesita y tendría un error.

Por último se puede observar el resultado del empaquetado en 3D utilizando el visualizador 3D de Proteus.

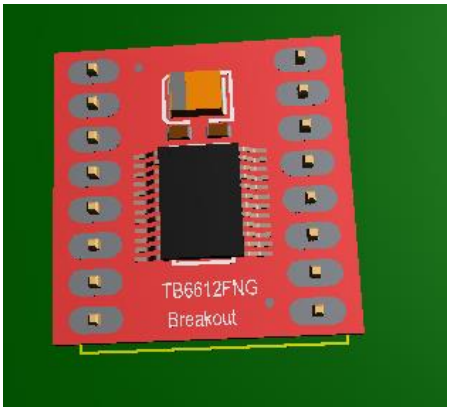


Imagen 22. 3D del empaquetado "Puente H TB6612FNG".

4.2.5 PULSADOR (2 pines)

Para realizar el pulsador se inicia colocando dos Pads de perforación a 200 mili pulgadas entre ambos Pads, posteriormente se realiza un recuadro que debe pasar por en medio de ambos Pads de perforación, la distancia de este recuadro será de 200 milipulgadas tal como la distancia entre ambos Pads. Una vez terminado el recuadro se colocara un círculo entre los Pads y el centro del recuadro, tal como lo muestra la imagen de referencia. Si se tiene una vista o cuadricula de 50 milipulgadas entonces el círculo quedara centrado prácticamente con solo colocarlo. Se tienen que enumerar ambos Pads de perforación, por tanto se procede a utilizar la herramienta "Automatic Name Generator" asignando el numero 0 al Pad derecho y el 1 la Pad izquierdo.

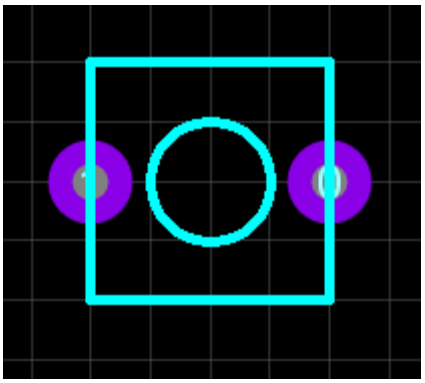


Imagen 23. Pulsador.

Ahora concluido este proceso se tienen que seleccionar todas las líneas del pulsador y dando click derecho sobre el pulsador se debe seleccionar la opción "Make Package" para llenar los campos correspondientes del empaquetado para el pulsador, tal como se hizo con el puente H o el Arduino Nano.

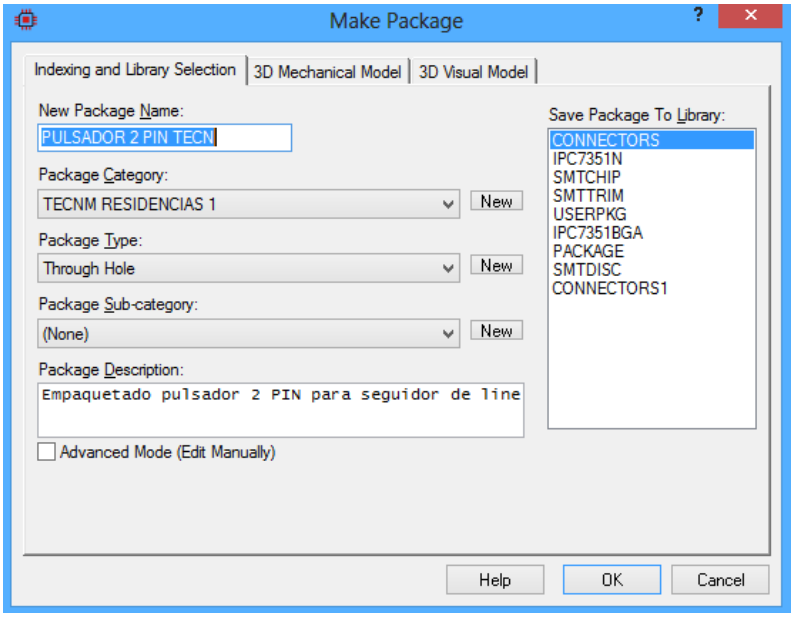


Imagen 24. Empaquetado del pulsador de dos pines.

En la parte superior "3D Visual Mode" se debe dar click en la opción "M-CAD (STEP or IGES) File, y nuevamente click en la figura de folder para ser enviado a la carpeta M-CAD que contiene las librerías de Proteus, ahí se tiene que seleccionar la imagen del pulsador de 2 pines que previamente fue colocado.

La imagen debe quedar en dirección a las perforaciones que se muestras en la visualización 3D, para ello se debe desplazar el pulsador sobre 3 ejes, X, Y y Z tanto en Rotation como en Offset, incrementando o disminuyendo los valores del empaquetado que debe de ser ubicado en la posición correcta, es importante que el pulsador quede en la misma dirección y sentido que las líneas amarillas que están dibujadas en la parte inferior de la placa ejemplo.

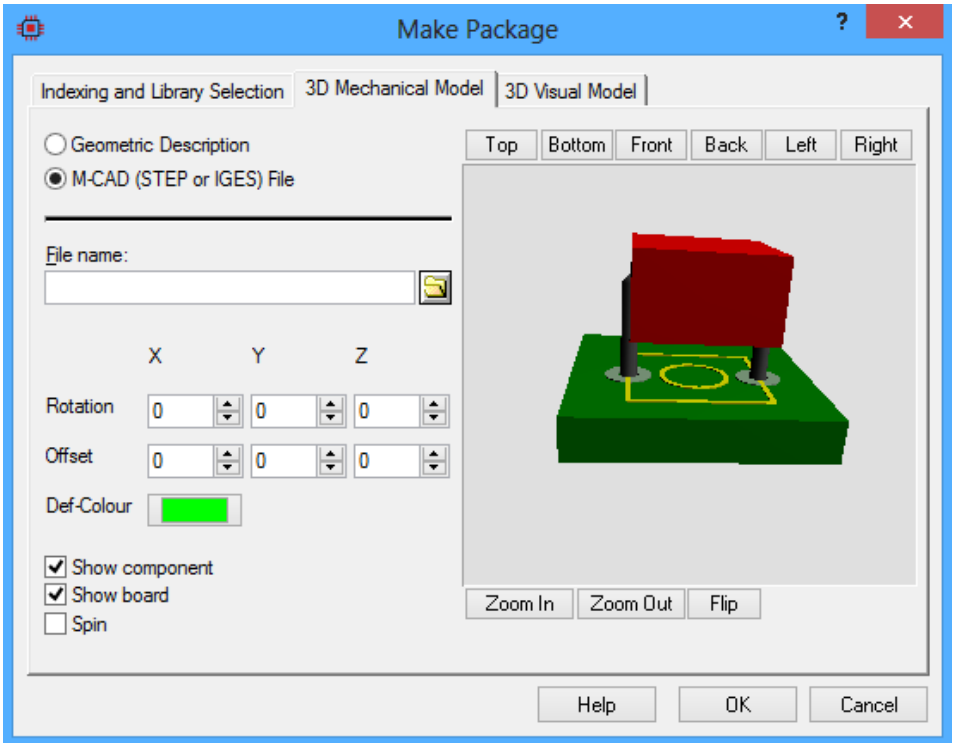


Imagen 25. Visual Mode.

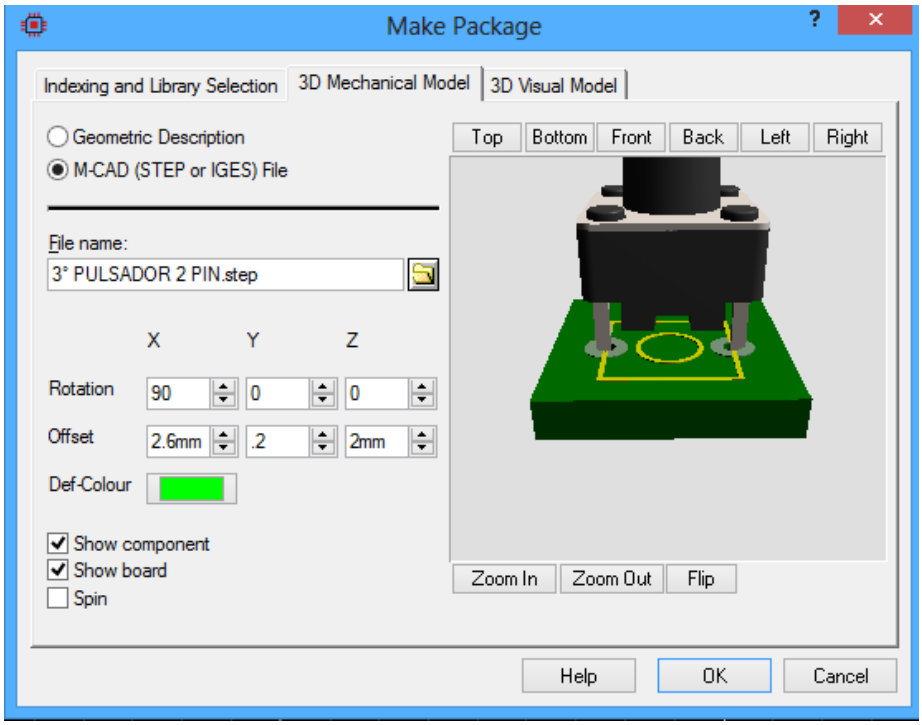


Imagen 26. Visual Mode con imagen cargada.

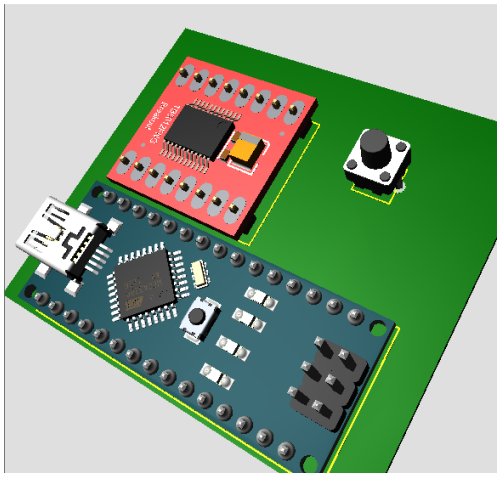


Imagen 27. 3D del empaquetado “Pulsador de 2 pines”.

4.2.6 Espadines (11 y 3)

Debido a que Proteus no cuenta con espadines de 11 y de 3 se tienen que realizar sus empaquetados. Para comenzar el proceso se realizara el empaquetado del espadín de 11. Para crear el de 11 en la parte de herramientas ubicada en la parte izquierda de la pantalla se tiene que seleccionar Package Mode y dar click en la P azul para desplegar el buscador de componentes de Proteus, una vez ahí se escribirá en el buscador “CONN-SIL” y se podrán ver todas las opciones de espadines que tiene por defecto Proteus en sus librerías.

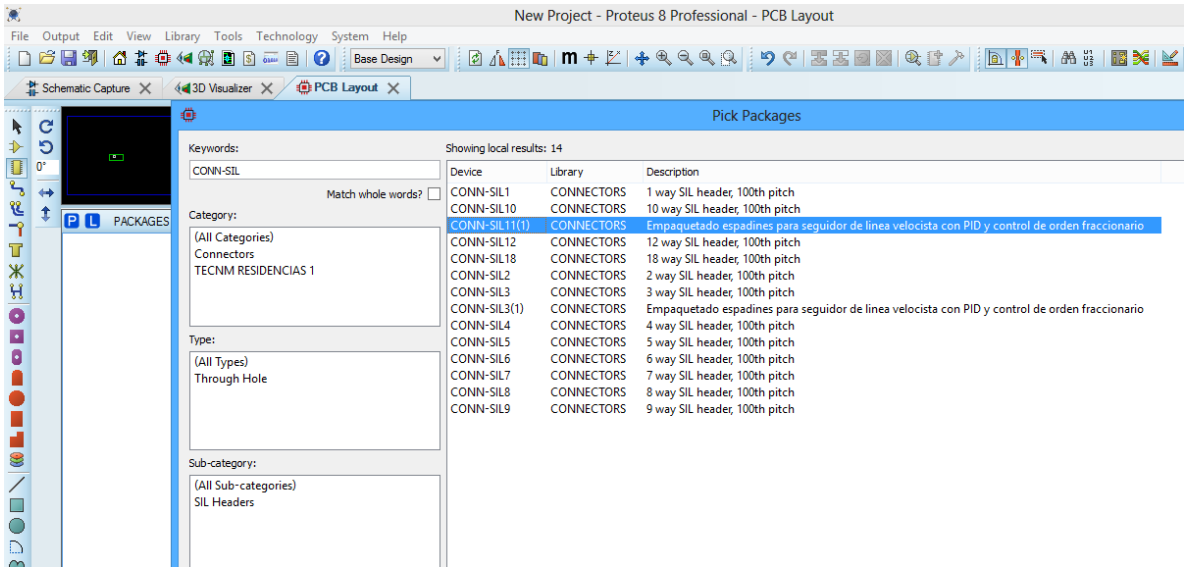


Imagen 28. Buscador de componentes de Proteus.

Debido a que ya se han realizado con anterioridad estos componentes para el proyecto, en el buscador se pueden observar los espadines requeridos, sin embargo, cuando no se tienen estos espadines el buscador no va a mostrarlos, por ello se seleccionara el espadín de 10 para realizar el de 11. Una vez seleccionado el espadín de 10 se debe dar click en aceptar para poder colocarlo en el espacio de trabajo, después se dará click derecho sobre el componente, y se seleccionara la opción “Descompose Tagged Objects” que desplegara palabras en azul debajo del componente, las cuales se tienen que seleccionar y eliminar.

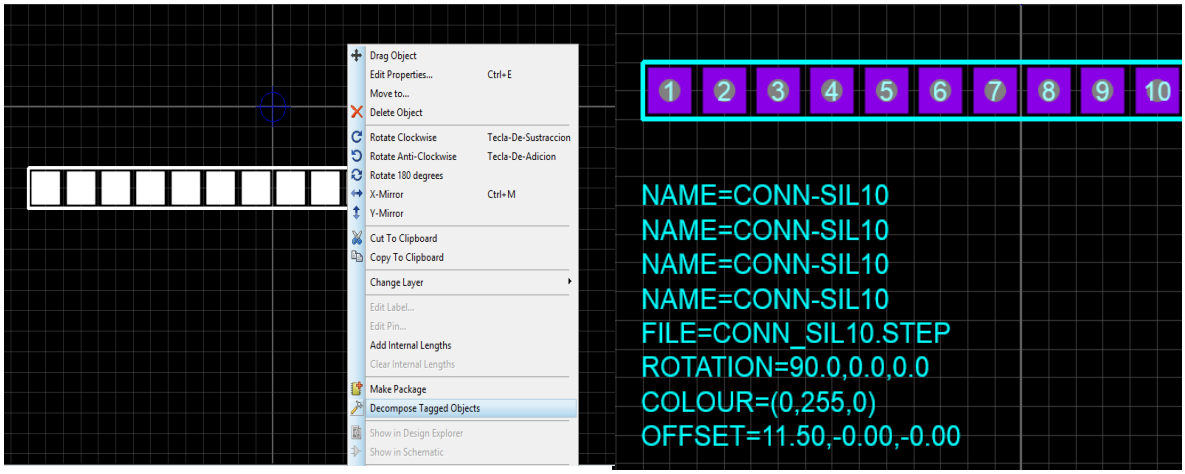


Imagen 29. Descomposición de componentes.

Una vez eliminada la información del componente se tiene que eliminar el recuadro azul que contienen los Pads enumerados, para copiar el Pad número uno pegarlo a lado izquierdo y cambiar el número uno por cero, dando doble click sobre el Pad se puede cambiar el número a 0.

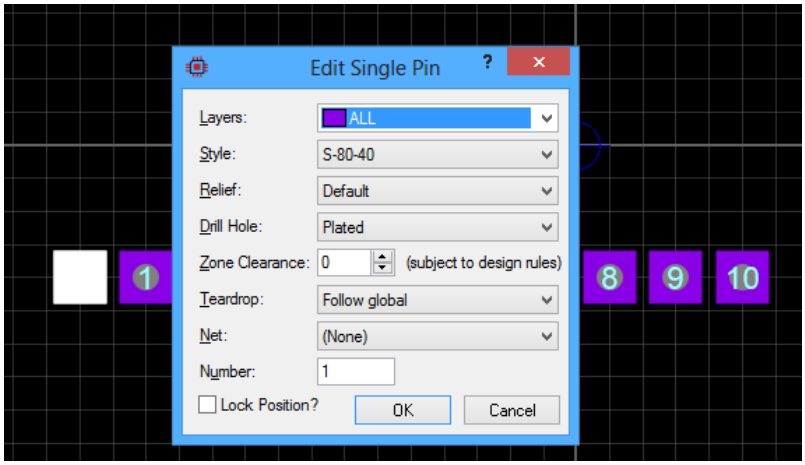


Imagen 30. Cambio de número de pad.

Por último se deben encapsular los 11 Pads de perforación y realizar el llenado de datos del empaquetado, nuevamente seleccionando todas las líneas del espadín se tiene que dar click derecho, seleccionar la opción “Make Package” y realizar el llenado de datos.

Imagen 31. Espadines de 11.

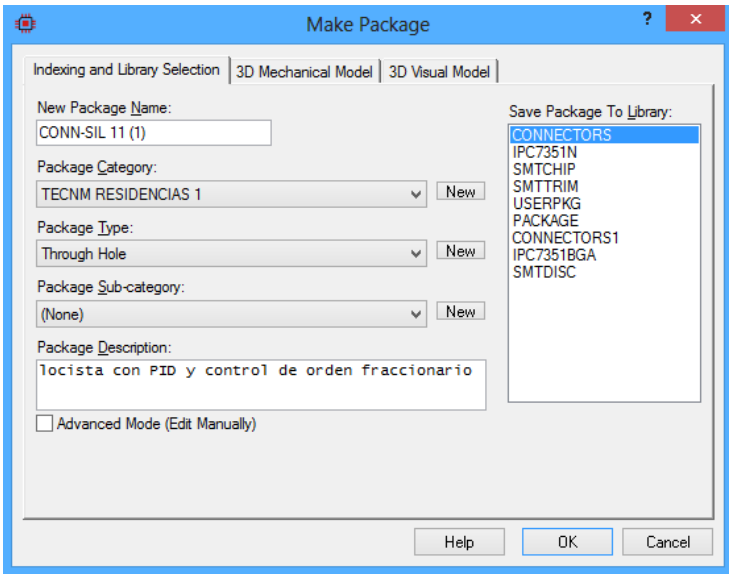


Imagen 32. Empaquetado de espadines de 11.

Debido a que no se tiene imagen de los espadines no es necesario buscarla ni cargarla en el “3D Visual Mode” lo único que se deberá hacer es, colocar en la posición correcta a los espadines con las perforaciones de la placa ejemplo.

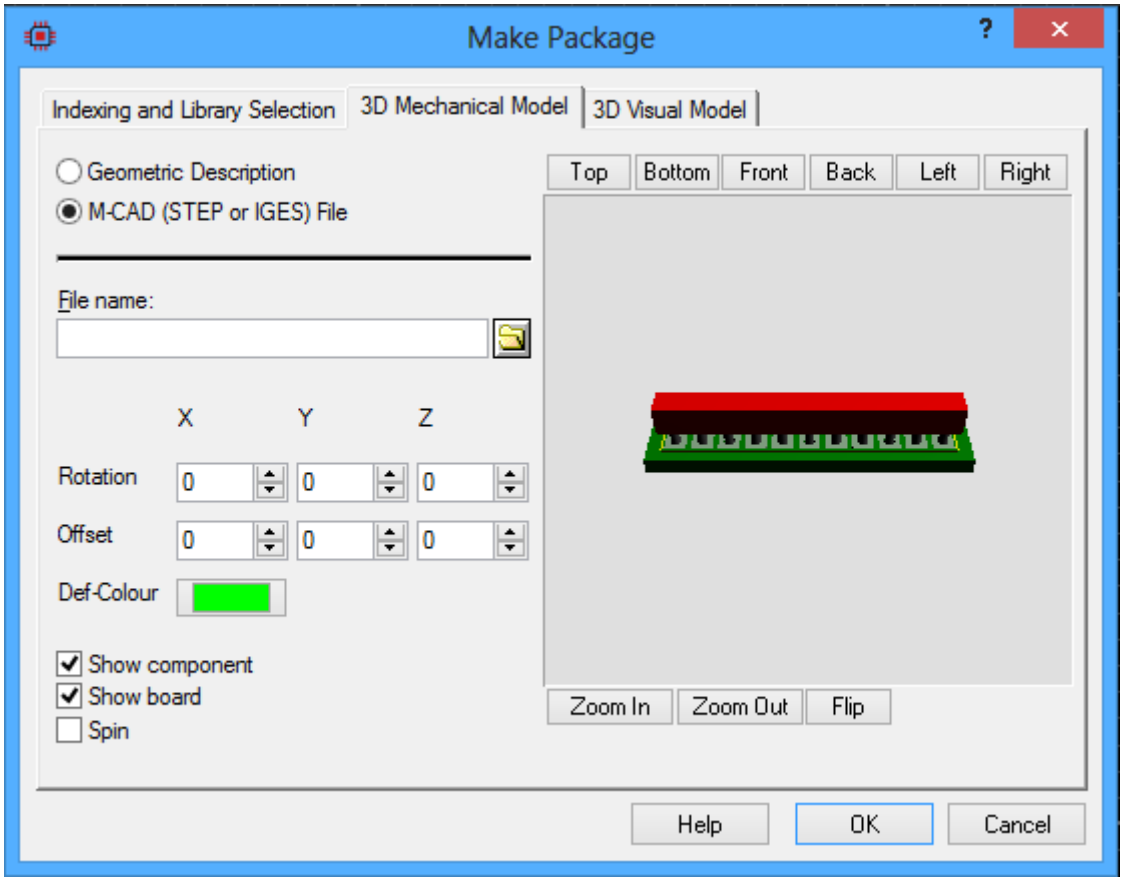


Imagen 33. Visual Mode.

El proceso para el empaquetado del espadín de 3 es totalmente el mismo proceso que con el espadín de 11, ahora bien, también puede existir la opción en que por defecto se encuentre al espadín de 3 justo en el buscador de componentes, que en dado caso, este proceso no tendría que ser realizado.

Para crear el espadín de 3 en la parte de herramientas se seleccionara la opción Package Mode > click en la P azul que desplegará el buscador de componentes de Proteus, una vez ahí se tiene que escribir en el buscador “CONN-SIL” que desplegara todas las opciones por defecto que tiene Proteus en sus librerías. Seleccionando al espadín de 4 se debe dar click en “aceptar” para insertar al espadín en el espacio de trabajo. Posteriormente se procede a dar click derecho sobre el espadín

de 4 y seleccionar la opción “Descompose Tagged Objects” para eliminar las letras azules debajo del componente.

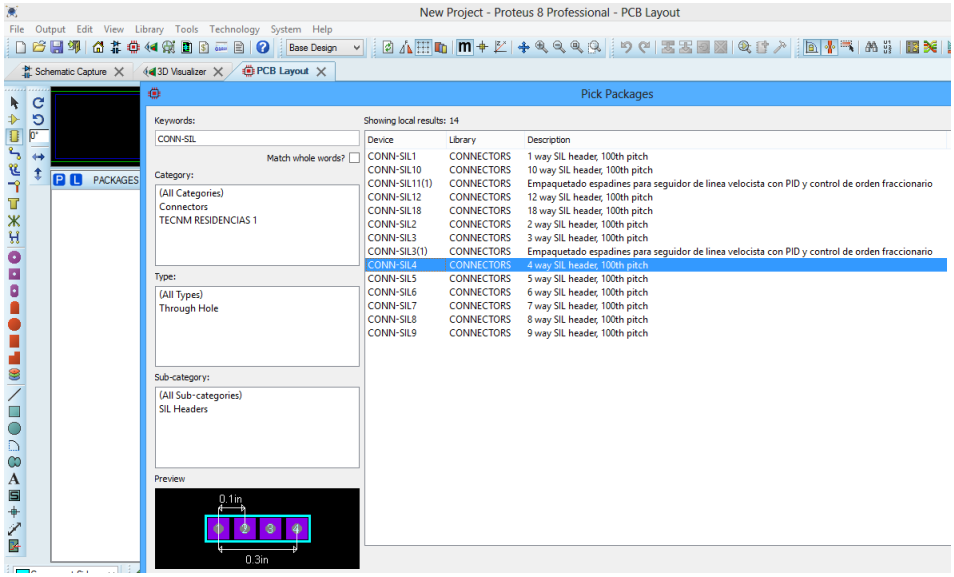


Imagen 34. Selección espadín de 4.

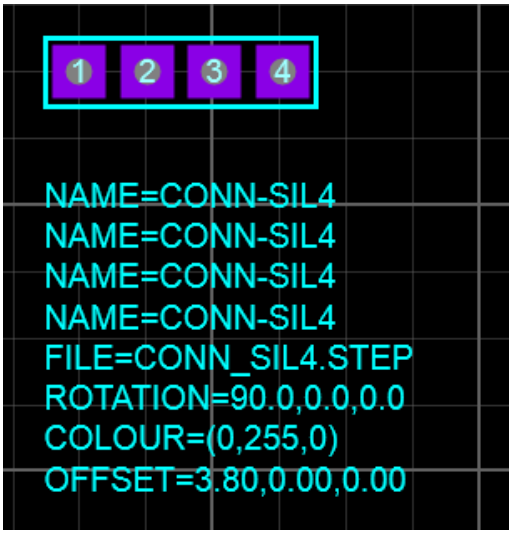


Imagen 35. Descomposición del componente.

Una vez eliminada la información del componente se debe eliminar el recuadro azul que encapsula a los Pads de perforación para eliminar un Pad. Después se tiene que reenumerar el orden en el que están los números mostrados, no partiendo del 1 sino del 0. Nuevamente dando doble

click izquierdo sobre el número, se arrojara la venta para editar, color, número, etc., para así cambiar el orden de 1, 2, 3 por 0, 1, 2 y re-encapsular los Pads de perforación con un rectángulo, para proceder a hacer el empaquetado.

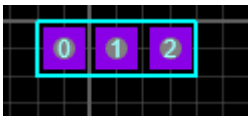


Imagen 36. Espadín de 3 pines.

Por último se realiza el llenado de datos del empaquetado, nuevamente seleccionando todas las líneas del espadín se debe dar click derecho y seleccionar la opción “Make Package”.

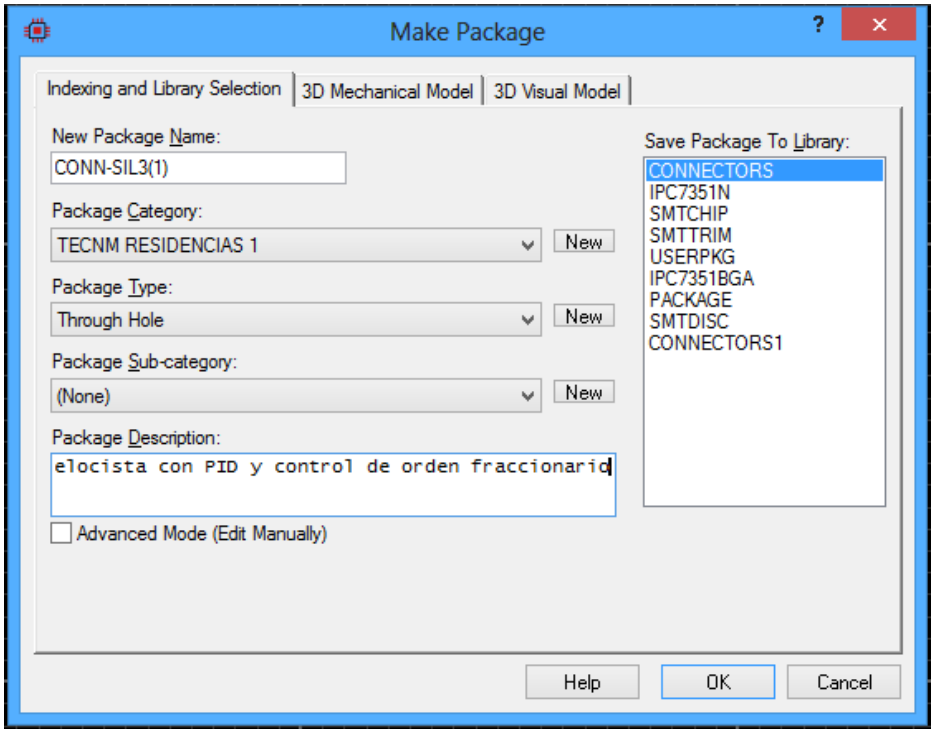


Imagen 37. Espadín de 3 pines.

Debido a que no se tienen imagen de los espadines esta no se buscara, ni cargara en el “3D Visual Mode” lo único que se deberá hacer es colocar en la posición correcta a los espadines con las perforaciones de la placa ejemplo.

En la siguiente imagen se tiene un ejemplo grafico de cómo se ven los espadines sin la imagen cargada, en donde únicamente se verá en color rojo todo el modelado 3D de los espadines. Después se puede observar en el visualizador 3D todos los componentes incluidos los espadines de 3 y de 11 en la placa de ejemplo.

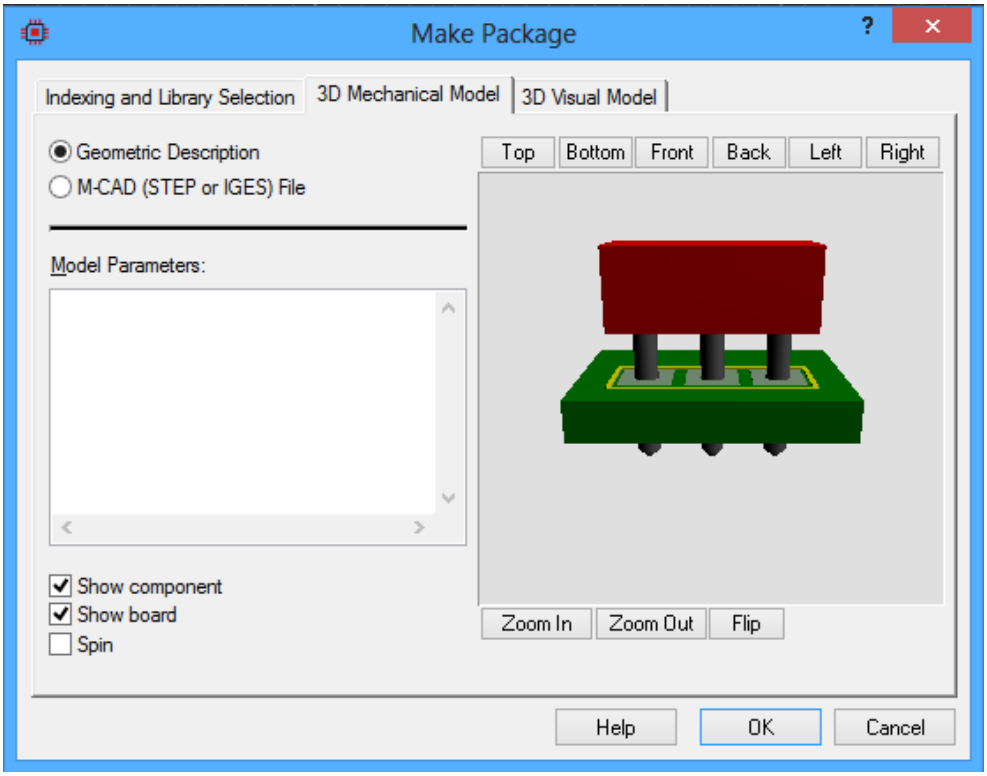


Imagen 38. Visual Mode.

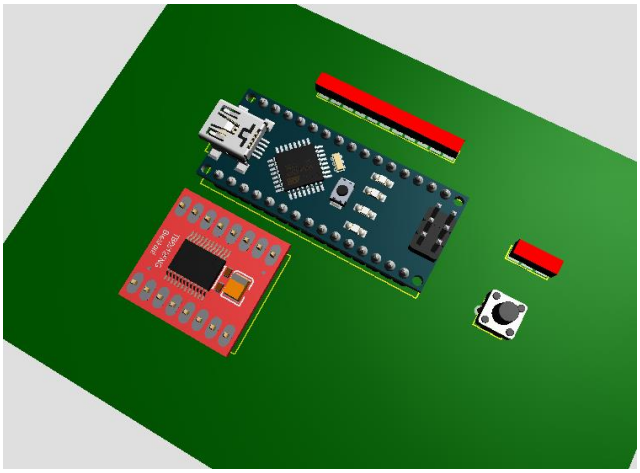


Imagen 39. 3D del empaquetado “Espadines de 3 y de 11”.

4.2.5 Switch

Para la creación del Switch se toma como punto de partida el espadín de 3, que debe ser seleccionado en el buscador de componentes, al cual se le ha asignado el nombre de “CONN-SIL 3”. Una vez seleccionado, se tiene que dar click en aceptar y debe ser colocado en el espacio de trabajo.

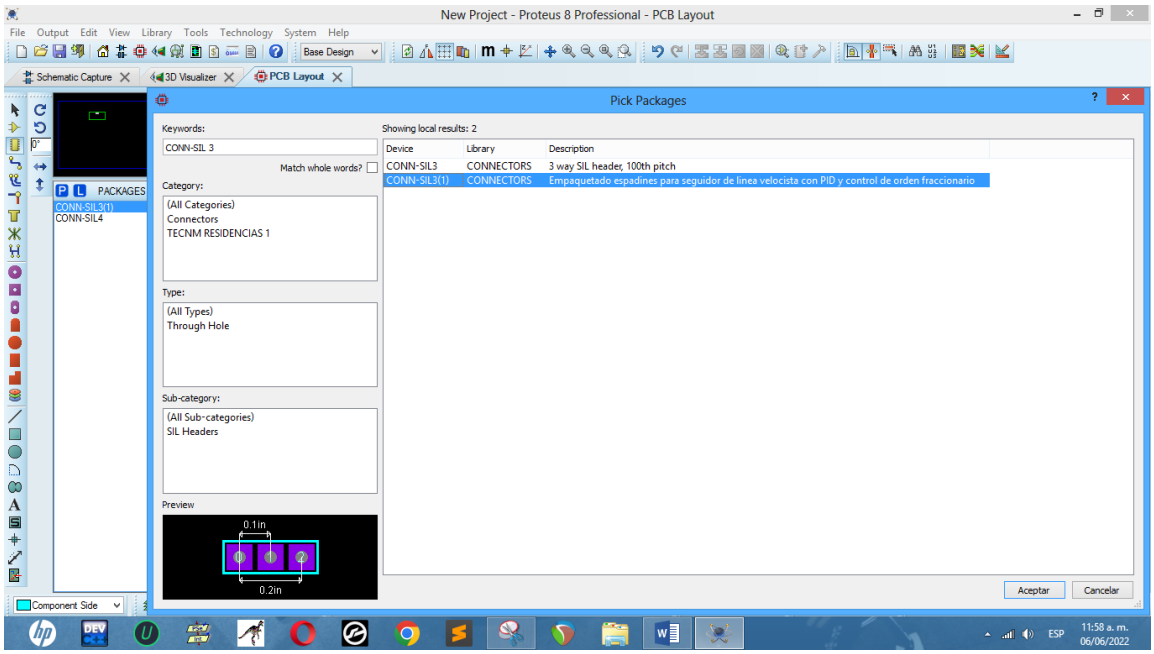


Imagen 40. Selección espadín de 3.

Ahora, se procede a seleccionar todo el componente y a realizar un nuevo empaquetado partiendo del espadín de 3, para ello se tiene que seleccionar la opción “Make Package” y llenar la nueva información que requiere el Switch con los datos siguientes (cabe mencionar que el nombre y la información de descripción del Switch es lo que cambia):

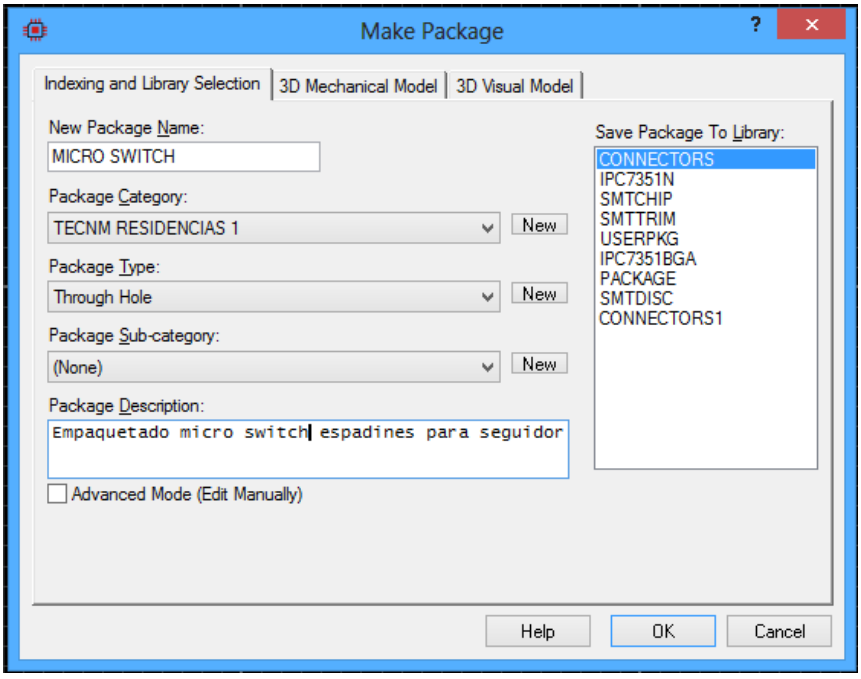


Imagen 41. Datos para el empaquetado del Switch.

Una vez concluido el llenado de datos, se tiene que ir al "3D Visual Model" para cargar la imagen del Switch. Se dará click en la opción "M-CAD (STEP or IGES) File, click en la figura de carpeta para ser enviado a la carpeta M-CAD de Proteus, donde se seleccionara la imagen del Switch de 3 pines previamente colocado y el empaquetado de la figura.

La imagen debe quedar en dirección a las perforaciones que se muestran en la visualización 3D, para ello nuevamente se desplazará el pulsador sobre 3 ejes, X, Y y Z tanto en Rotation como en Offset, incrementando o disminuyendo los valores para ubicar al empaquetado en la posición correcta. Es importante que el Switch quede en la misma dirección y sentido que las líneas amarillas que están dibujadas en la parte inferior de la placa ejemplo.

En la imagen siguiente se pueden observar las coordenadas utilizadas para empalmar el Switch con la placa de referencia mostrada, se pueden tomar esas coordenadas como ejemplo para que los pines y las perforaciones queden ubicadas en el lugar correcto y así no exista conflicto entre ellas.

Nota: nuevamente es importante no dar click en “ok” hasta tener el Switch ubicado de forma correcta sobre la placa para evitar que el empaquetado funcione de forma correcta.

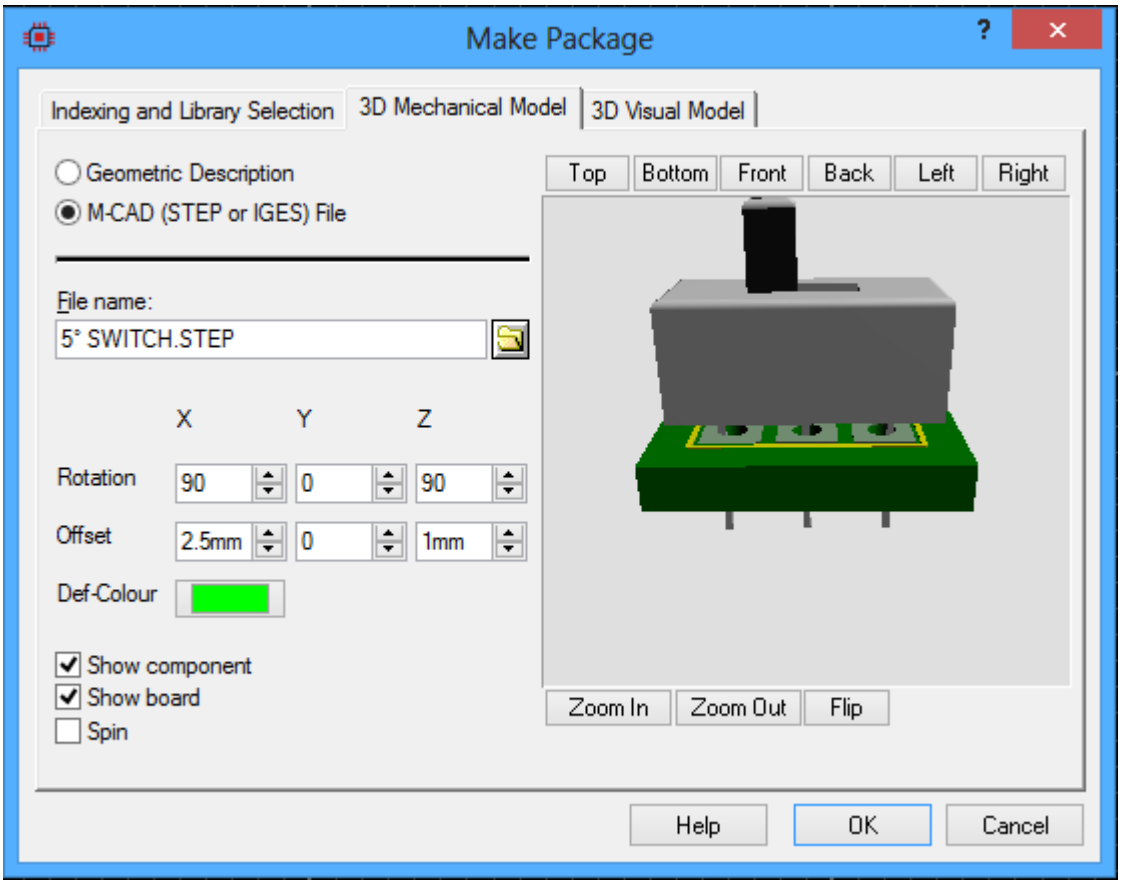


Imagen 42. Empaquetado del Switch.

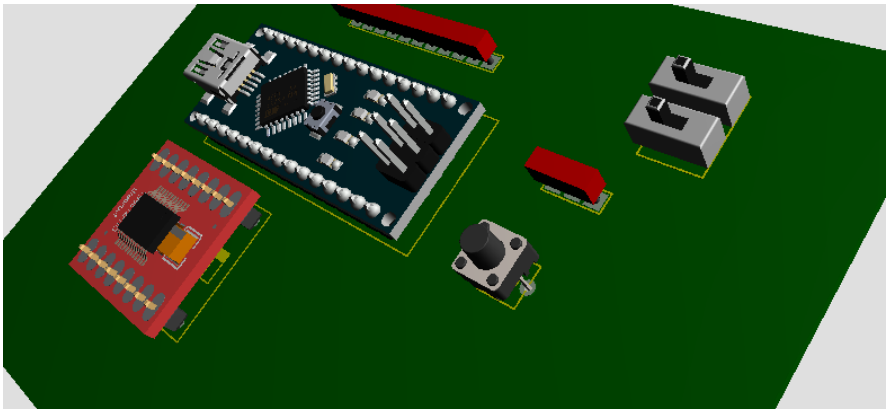


Imagen 43. 3D del empaquetado del Switch.

4.2.5 Micromotor tipo pololu

Para realizar los motores es muy importante tener en cuenta las medidas que se muestran a continuación, también se pueden consultar las medidas de los Micromotores tipo Pololu en internet, así la referencia de medición es a consideración de cada uno. Entonces en un principio se parte de la unidad de medida inglesa con una vista de 50 th. Yendo al apartado de herramientas en el lado izquierdo en "2D Graphic Circle Mode" se seleccionara el Pad M3, siendo el Pad de perforación donde se ubicarán los tornillos encargados de asegurar a los motores junto con los Brackets, que serán ubicados en el espacio de trabajo con una separación de 700 milipulgadas entre centros, después se colocaran los siguiente recuadros que se muestran a continuación con las siguientes medidas en milipulgadas. Cabe mencionar que los recuadros del espacio de trabajo están a 50 milipulgadas, por tanto al desconocer una medida se pueden contar los cuadros y sumarlos para conocer la medida requerida, pero solo en esta imagen del motor, las cuadriculas se pueden modificar a dependiendo de la necesidad del empaquetado, pasando de 50 milipulgadas a 25 o 5 a incluso 1 milipulgada.

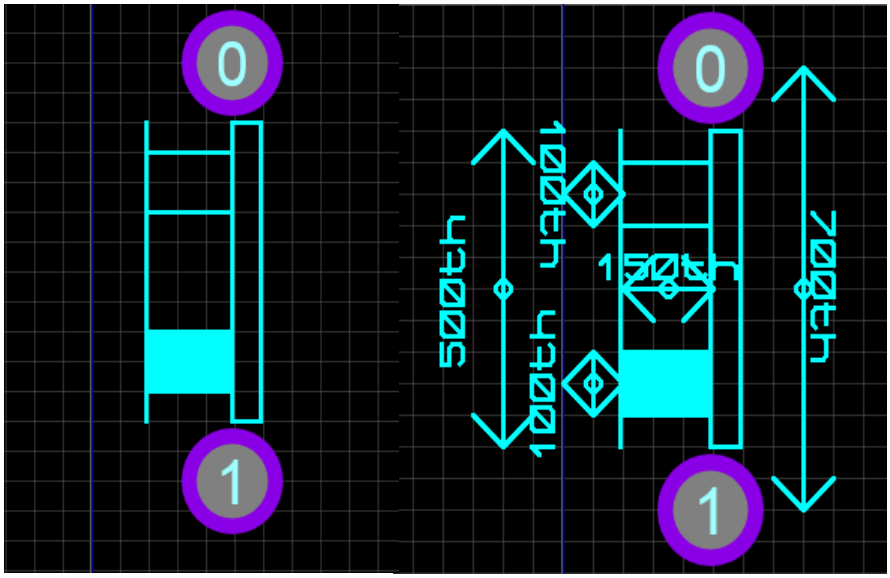


Imagen 44. Diseño de motores.

Una vez realizada la primer parte del motor, se deben colocar más recuadros en su mayoría utilizando la cuadricula de 25th en la visión de pulgadas, para ello se deben tomar como referencia las medidas proporcionadas a continuación. En la parte izquierda se pueden observar las nuevas medidas utilizadas y en la parte derecha el avance del motor.

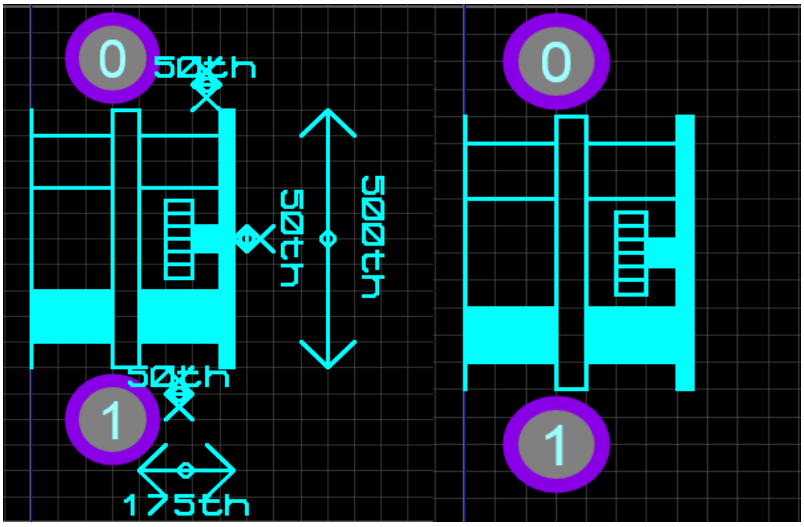


Imagen 45. Continuación de motores.

Por último se pueden observar las medidas de los últimos recuadros a utilizar para el diseño de los motores, además de dos Pads de perforación C90-40 agregados en la parte delantera de los motores, donde irán conectados dos pines machos para su alimentación, los Pads laterales traseros de mayor dimensión son las perforaciones que serán utilizadas para la inserción de tornillos que sujetarán los Brakets de los motores, teniéndolos fijos a la placa. Las medidas para la parte final de los motores se podrán visualizar en la siguiente imagen. Los recuadros mostrados están a 25 milipulgadas.

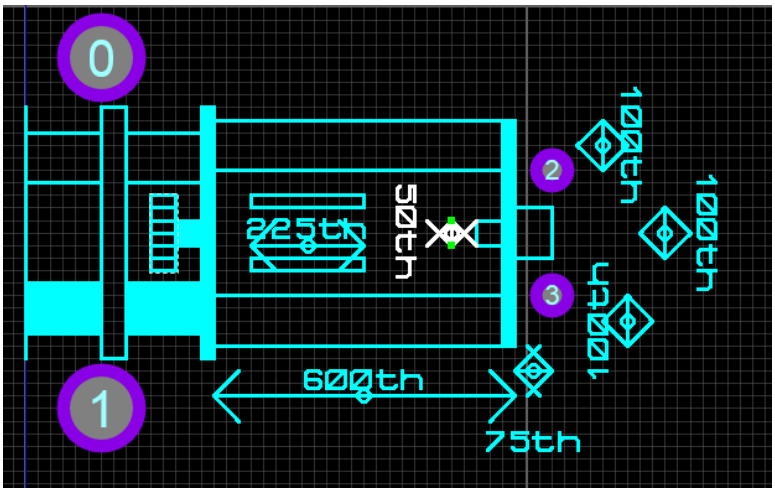


Imagen 46. Medidas finales de los motores.

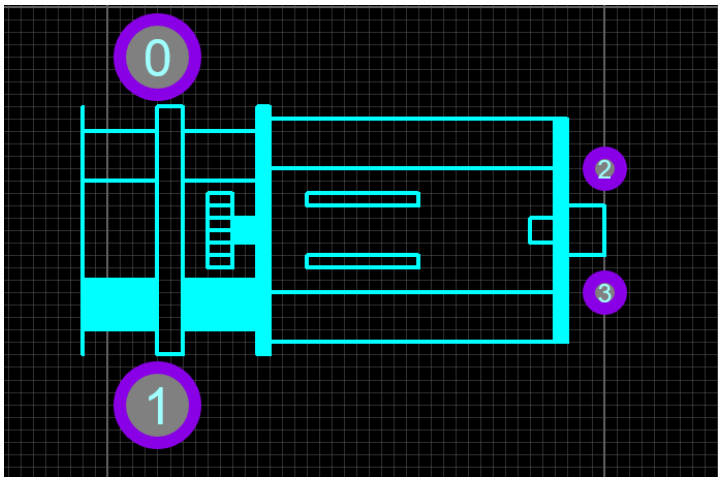


Imagen 47. Diseño de motores.

Concluido el diseño del motor lo siguiente a realizar es el empaquetado, seleccionando todo el diseño del motor se procede a dar click derecho y seleccionar “Make Package” para realizar el llenado de información del empaquetado, la siguiente imagen sirve como referencia del llenado de información necesaria para el empaquetado.

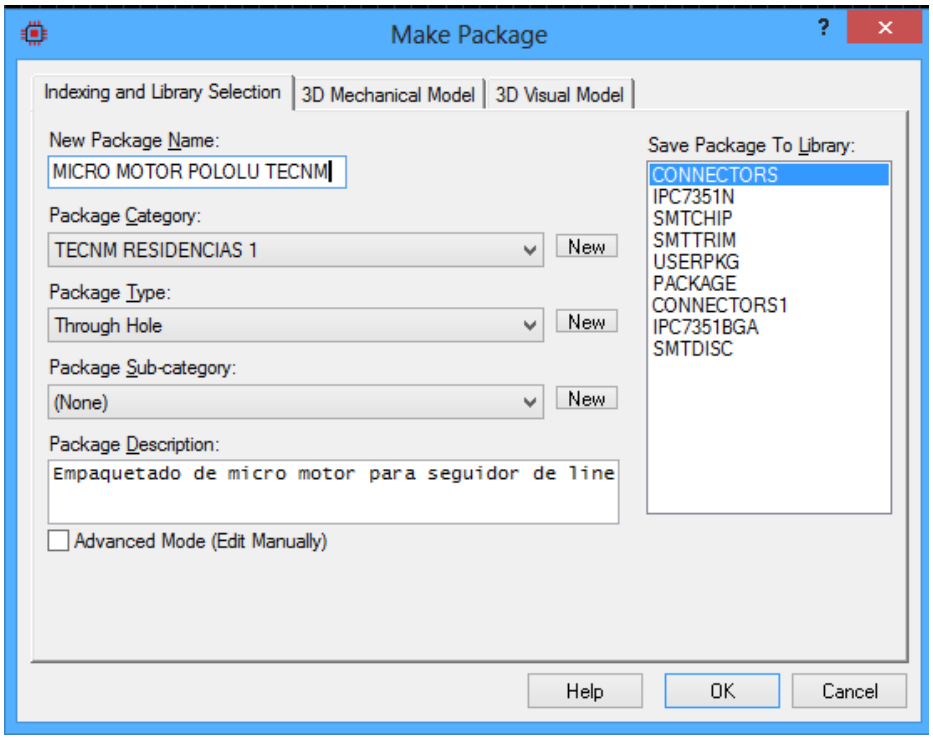


Imagen 48. Empaquetado de motores.

Posteriormente con la herramienta “3D Visual Model” se ingresa la imagen de los motores, junto con los Brackets y las llantas de caucho, para ello en File Name se tiene que dar click sobre la imagen de carpeta, que desplegara una ventana donde se tiene que seleccionar la imagen de los motores. Previamente se puede observar el molde en 3D de lo realizado hasta el momento, con las posiciones en los ejes X, Y y Z en valores de 0, puesto que no tiene una imagen cargada los motores están posicionados de forma correcta sobre la placa de ejemplo. Pero una vez agregada la imagen

se tiene que posicionar el motor de forma correcta en base a las perforaciones y la base en amarillo sobre la placa de ejemplo.

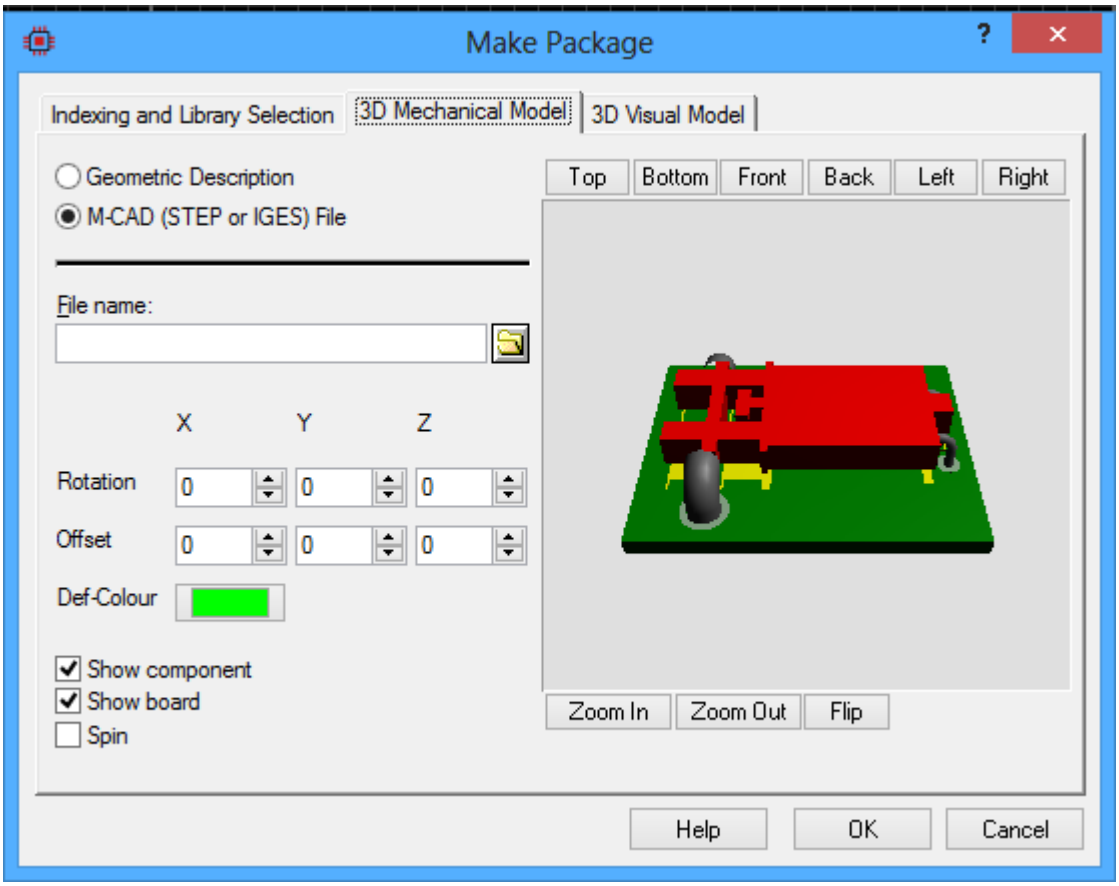


Imagen 49. 3D Visual Mode.

Para concluir con el empaquetado, en el “3D Visual Mode” se puede visualizar la imagen en concreto de los motores, junto con los Brackets y la llanta, para este caso se tendrán que mover el motor, el Bracket y la llanta en los ejes X, Y, Z tanto en rotación como en offset para quedar alineados de forma correcta. Como referencia se pueden tomar los valores de los 3 ejes mostrados a continuación pero es importante revisar que la posición coincida de forma exacta con la posición asignada por cada uno, por eso se debe tener en cuenta el diagrama resultante del diseño del motor

que se puede apreciar en la placa de ejemplo. Después en una segunda imagen se pueden observar todos los componentes utilizados y realizados hasta el momento.

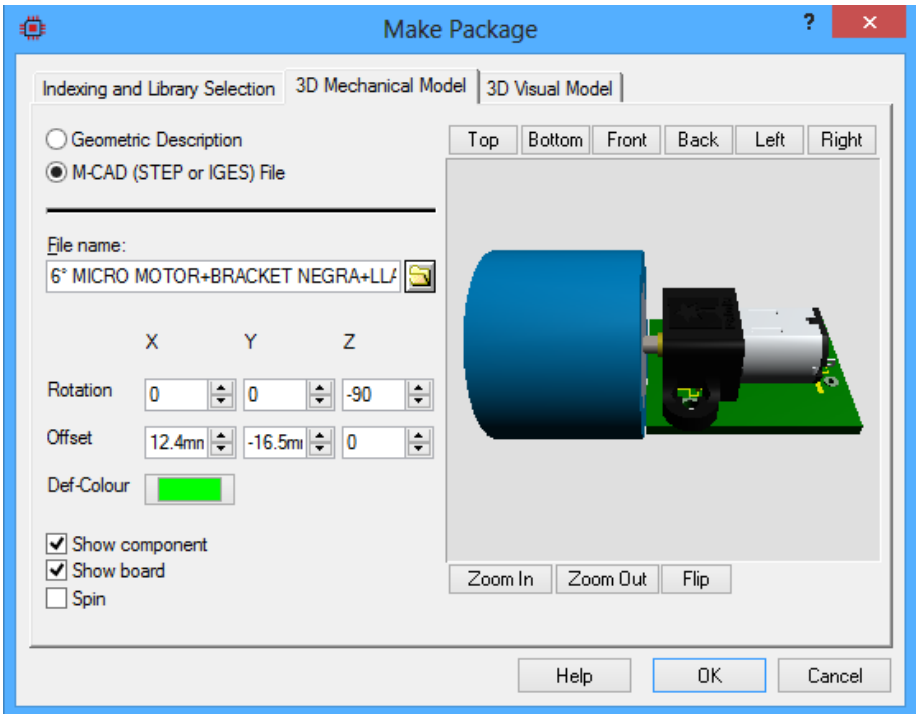


Imagen 50. Motor tipo pololu.

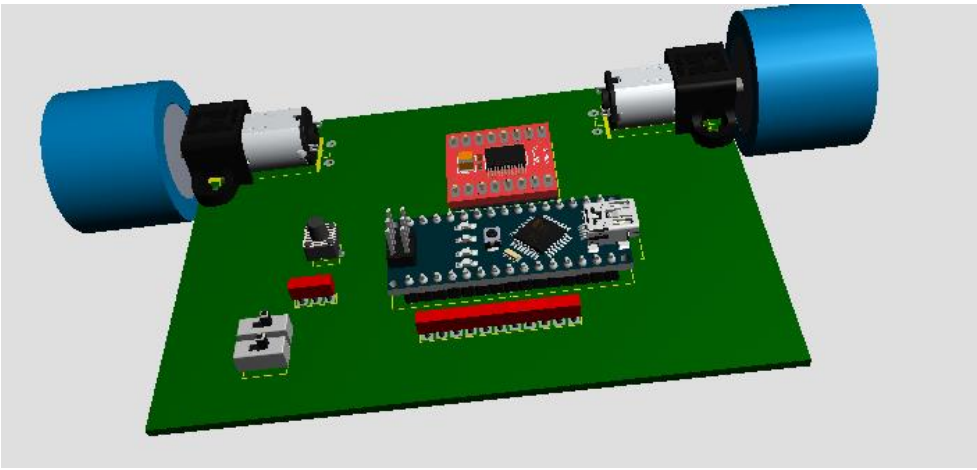


Imagen 51. 3D Motores en placa.

4.2.6 Turbina

Para realizar la turbina el proceso es bastante sencillo puesto que la placa para la turbina física solo debe tener la misma medida que los Pads de perforación, en este caso serán utilizados los Pads C500-60. Se debe colocar un solo Pad sobre la superficie de trabajo, no importa el lugar, pudiendo ser cualquier zona, para después comenzar con el empaquetado.

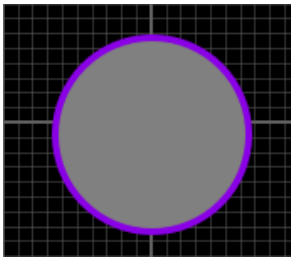


Imagen 52. Pad C500-60 para turbina.

Una vez colocado el Pad se debe dar click derecho sobre él, y seleccionar la opción "Make Package" para poder llenar los datos correspondientes al empaquetado de la turbina. En la siguiente imagen se pueden observar los datos utilizados para la turbina que de igual manera que con los empaquetados anteriores se puede utilizar como referencia.

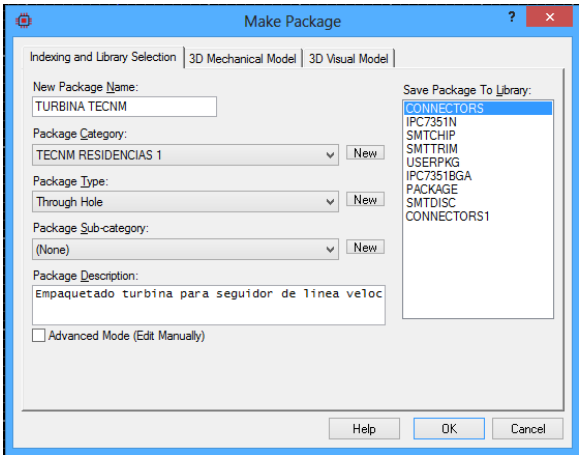


Imagen 53. 3D. Datos del empaquetado de turbina.

Para finalizar con el empaquetado en necesario utilizar la herramienta “3D visual mode” donde se seleccionara la imagen correspondiente a la turbina, la cual ira montada sobre la placa. En un principio no se visualizara la turbina, es por ello que se debe agregar la imagen para tenerla representada sobre la placa de ejemplo. En la parte de File Name se tiene que seleccionar el icono de carpetas que desplegara la carpeta donde están todas las imágenes de los empaquetados, seleccionando la correspondiente a la turbina, se da click en aceptar e inmediatamente se tendrá la imagen montada. Lo único restante será posicionar de forma adecuada la turbina por encima de la placa de ejemplo del “3D Visual Mode”.

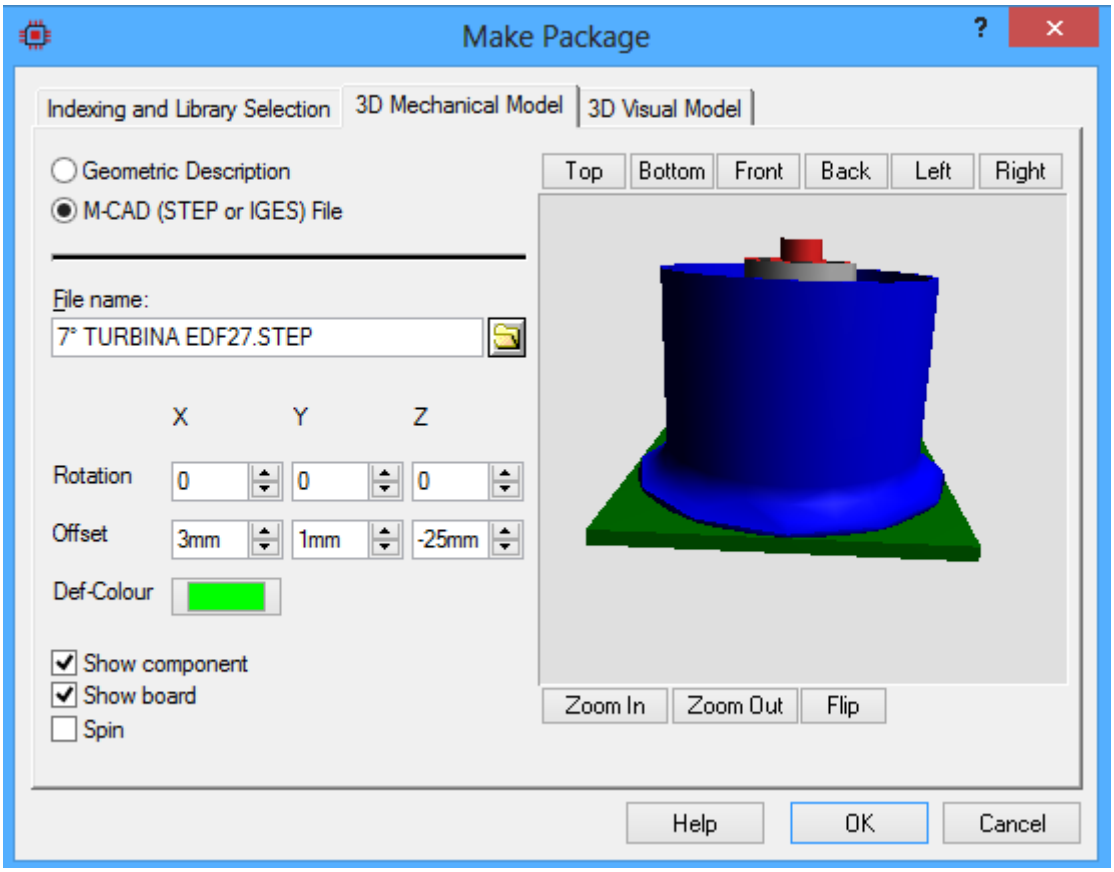


Imagen 54. 3D Turbina.

Por último, para tener una imagen general en 3D de todos los empaquetados deben ser insertados en el área de trabajo, para ello en la parte izquierda de la pantalla, justo en las herramientas de arriba hacia abajo se tiene que seleccionar la opción “Package Mode” ahí estarán todos los componentes o empaquetados realizados hasta el momento. Seleccionando uno por uno deberán ser colocados en el espacio de trabajo para colocarlos de tal modo en que los motores queden a los extremos, la turbina al centro, el Arduino Nano de lado izquierdo y el puente H a lado derecho, el espadín de 11 en la parte superior, dos Switch a su derecha, el botón en la parte inferior izquierda y el espadín de 3 bajo los Switch (se puede recurrir a la imagen siguiente). Una vez colocados los componentes, en el “3D Visualizer” se pueden observar pero sin la placa, estando en orden pero volando en el vacío, para poder visualizar los componentes en una placa de ejemplo, se tiene que encapsular a dichos componentes en un rectángulo, por tanto seleccionando la opción de rectángulo se puede realizar esta acción, pero con la diferencia de que en la parte inferior no se usara la opción Top Silk, dando click ahí mismo y se debe seleccionar la opción Board Edge, como referencia se puede observar que la opción Top Silk tiene un color azul suave, mientras que el Board Edge tiene un color amarillo. Siguiendo todos los pasos, la placa con los empaquetados se verá de la siguiente manera:

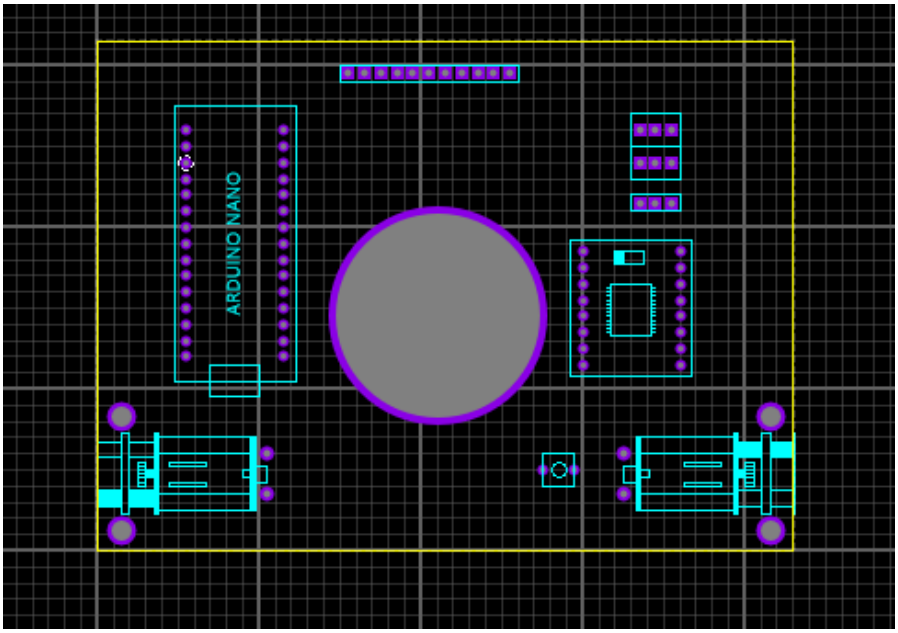


Imagen 55. Placa con empaquetados.

Usando el visualizador 3D la placa con empaquetados se verán de la siguiente manera:

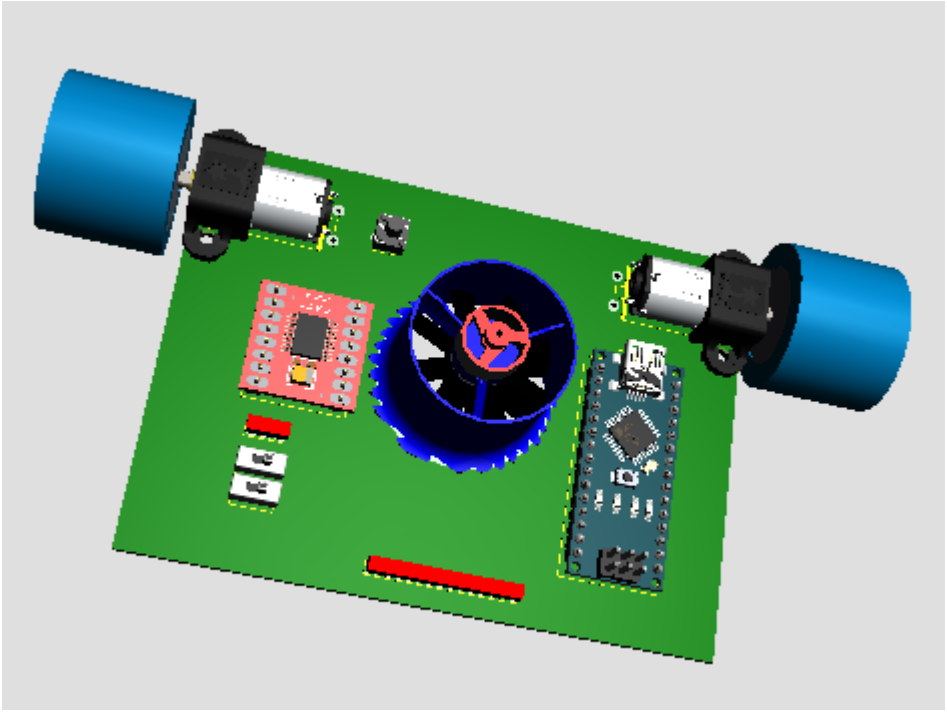


Imagen XX. Placa con empaquetados 3D.

4.3 Construcción de dispositivos en proteus (schematich capture)

Lo siguiente es realizar los dispositivos de los empaquetados en el apartado Schematich Capture de Proteus, se realizaran todas las conexiones necesarias entre dispositivos para que todos los empaquetados tengan relación entre ellos.

4.3.1 Arduino nano

Se inicia realizando un recuadro con la herramienta de figuras ubicada a la izquierda de la pantalla, las herramientas se ubican en la misma dirección y sentido que en la pestaña de PCB Layout, una vez realizado el rectángulo, con la herramienta Device Pin Mode ubicada de igual forma en el lado izquierdo de la pantalla se deben colocar 15 pines en el rectángulo por cada lado, correspondientes al Arduino Nano. En la selección del tamaño de los pines se puede utilizar la opción Short para unos pines cortos.

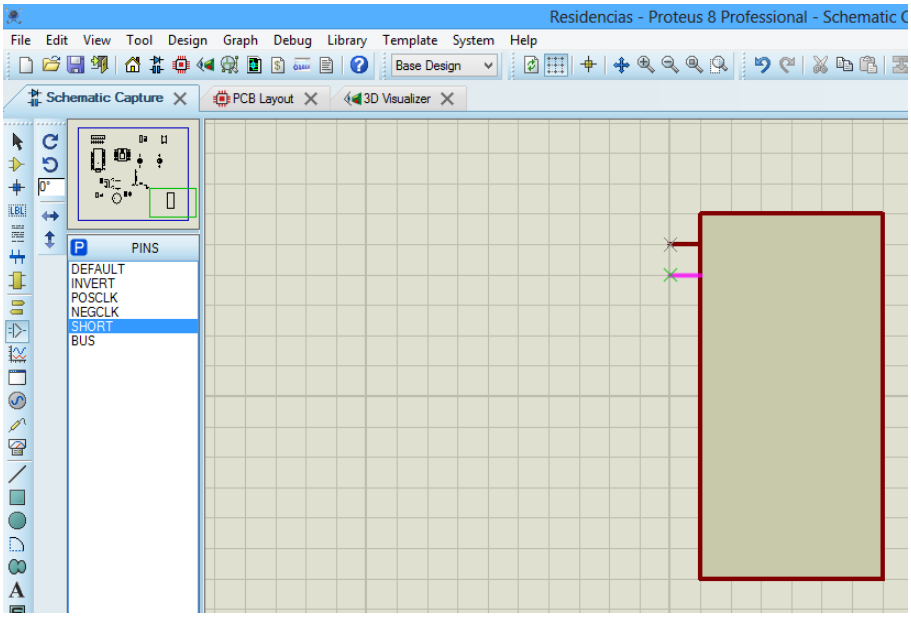


Imagen 57. Pines para Arduino.

En caso de que el rectángulo sea de menor tamaño con respecto a los pines colocados se deberá aumentar su tamaño de forma proporcional de tal modo que los pines queden de forma correcta en el espacio del rectángulo para el Arduino Nano. Ahora, los pines tienen marcada una x en su parte izquierda (en caso de colocar los pines de lado izquierdo del rectángulo), esa x debe quedar por fuera o en dirección contraria al rectángulo (lado izquierdo), por tanto sí los pines son colocados de esta forma para el lado derecho se tendrá un error, en este caso una opción es copiar toda la primer hilera de pines del lado izquierdo, pegarla al lado derecho, y con click derecho sobre ellas seleccionar la opción "X-mirror" para invertirlas, de este modo las x de los pines del lado derecho quedarán invertidas al rectángulo del Arduino Nano. Por último se tiene que colocar un recuadro en la parte inferior que haga alusión a la conexión USB del Arduino Nano, y con la herramienta de texto colocar el nombre "Arduino Nano" en el dispositivo.

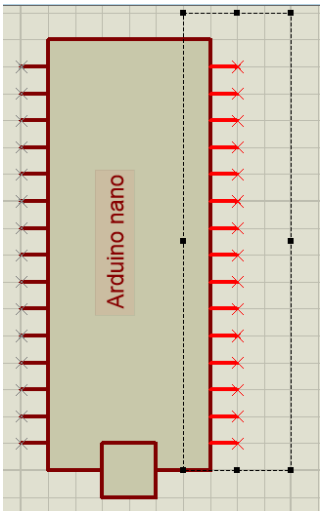


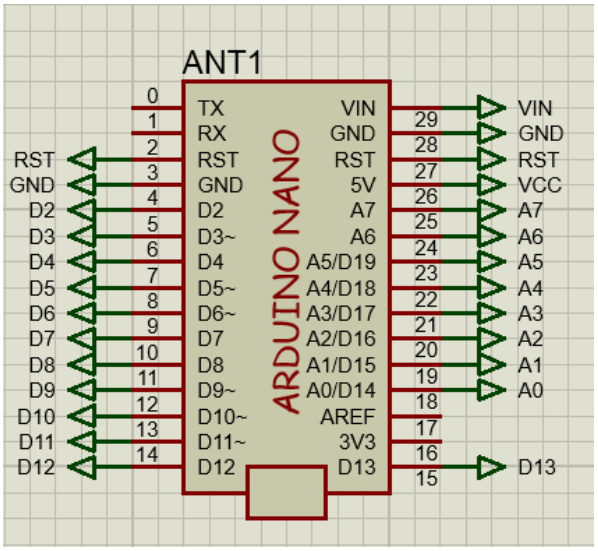
Imagen 58. Pines invertidos, y nombre del Arduino Nano.

Ahora lo único por hacer es colocar los nombres y números de cada uno de los pines tal cual fueron colocados en el empaquetado del Arduino Nano, partiendo del 0 en forma de U. Se puede

utilizar como referencia una imagen del Arduino Nano que contenga los nombres de cada uno de los pines.

Imagen 59. Nombre de los pines.

Para poder asignar un nombre a cada uno de los pines se debe dar doble click encima de cada uno, justo así se podrá asignar el nombre, número de pin y la información que deba mostrar el dispositivo, Tal como se muestra en la imagen siguiente haciendo alusión al pin número 0 del Arduino Nano.



También se deben marcar las 3 casillas debajo del número de pin que corresponden a cuerpo, nombre y número, para que se muestren en el dispositivo

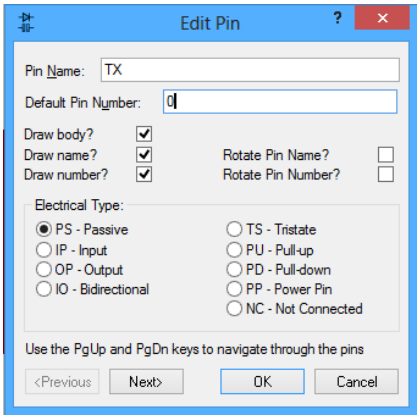


Imagen 60. Edición de pines.

Una vez llenado todo el dispositivo con los pines en el orden correcto se procede a seleccionar todo el dispositivo, para esto se tiene que dar click derecho sobre él, y seleccionar la opción "Make Device", saltará un recuadro indicando que existen pines con el mismo nombre, mensaje no importante que desaparece al dar click en aceptar pues sí existen dentro del Arduino Nano varios pines con el mismo nombre, que no afectan en la creación del dispositivo. Una vez realizado esto se puede comenzar a llenar el recuadro con de información

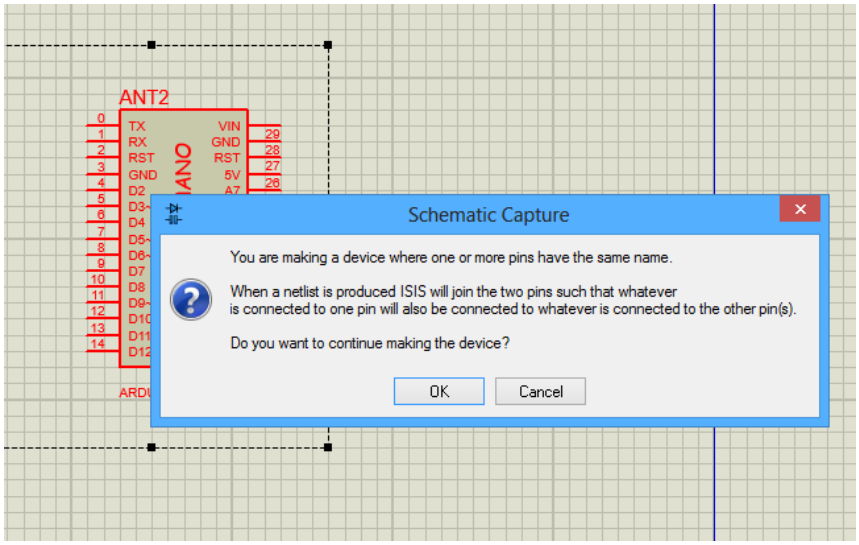


Imagen 61. Advertencia de mismos nombres.

Los pasos siguientes para la creación del dispositivo son varios, pero siguiéndolos de forma correcta, no habría ninguna complicación en la creación del dispositivo. Se inicia agregando el nombre del dispositivo junto con su preferencia.

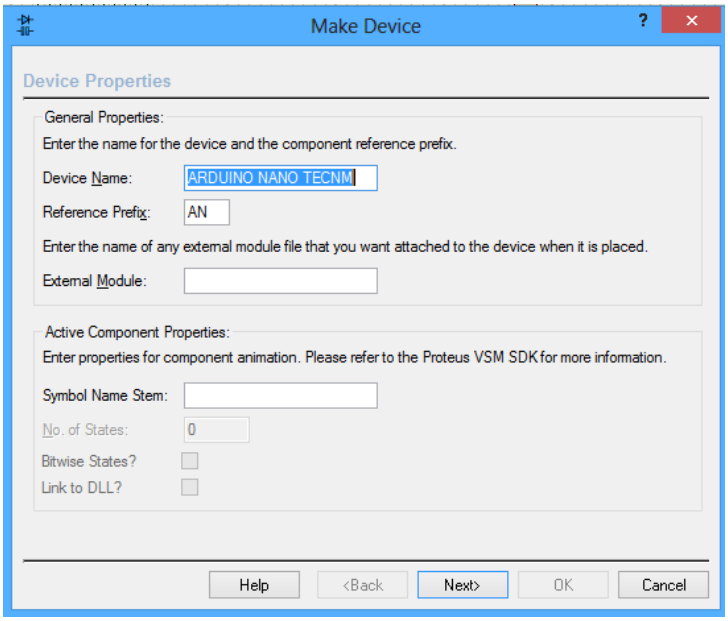


Imagen 62. Propiedades Generales.

Con los datos colocados se procede a dar click en Next para pasar a la siguiente pestaña, donde se tiene que agregar el empaquetado del Arduino Nano, para ello se debe seleccionar en la parte inferior izquierda del recuadro la opción Add/Edit, posteriormente en el nuevo recuadro en la parte superior se debe seleccionar Add para poder agregar el empaquetado. Una vez realizado este paso se utilizara el buscador donde únicamente se debe escribir Arduino Nano o el nombre con el cual se hizo el registro del empaquetado, seleccionarlo y dar click en Aceptar.

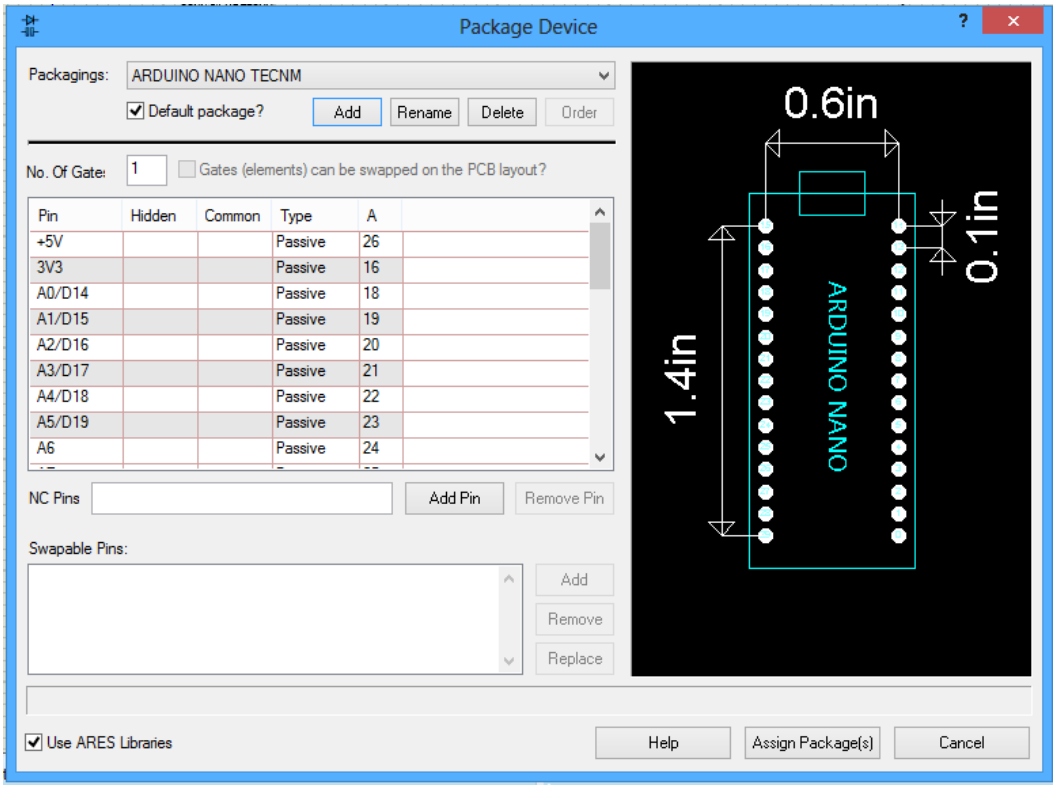


Imagen 63. Propiedades Generales.

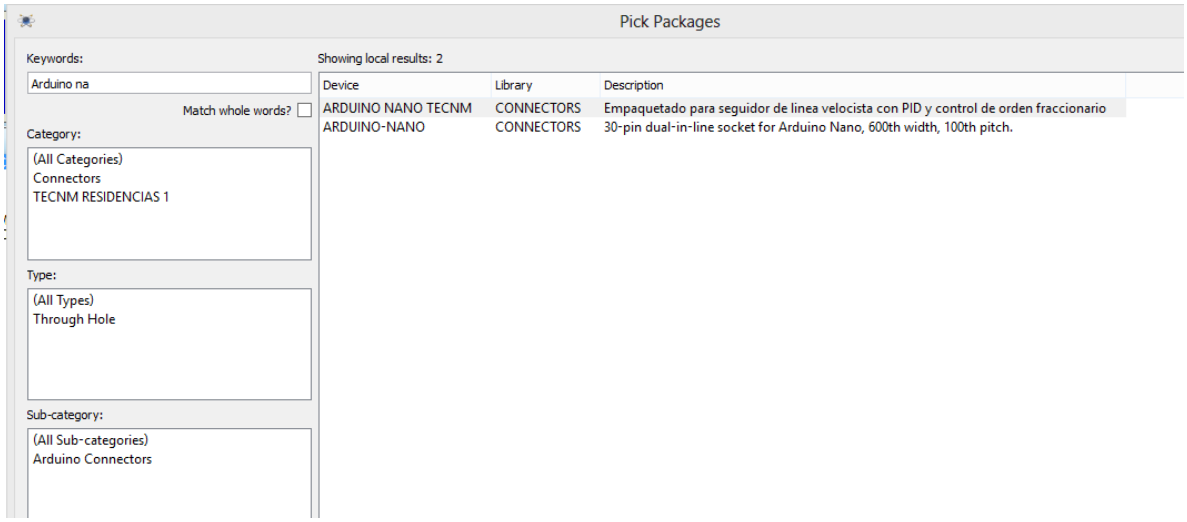


Imagen 64. Cargando empaquetado.

Ahora, con el Arduino Nano cargado se procede a la revisión de todos los Pads colocados en el empaquetado, deben estar marcados en color blanco, de lo contrario existirá un error al momento

de realizar el empaquetado. En la parte central izquierda se pueden encontrar las características de los pines, el tipo y la designación, en la parte central derecha se puede observar el empaquetado del Arduino Nano y todos los Pads junto con las medidas designadas, en caso de aparecer los Pads en color blanco el resultado será una creación del empaquetado correcta, si de lo contrario los Pads del empaquetado son color rosa, quiere decir que existe un error, que posiblemente sea un nombre o un orden equivocado y para resolverlo se tendría que editar el empaquetado.

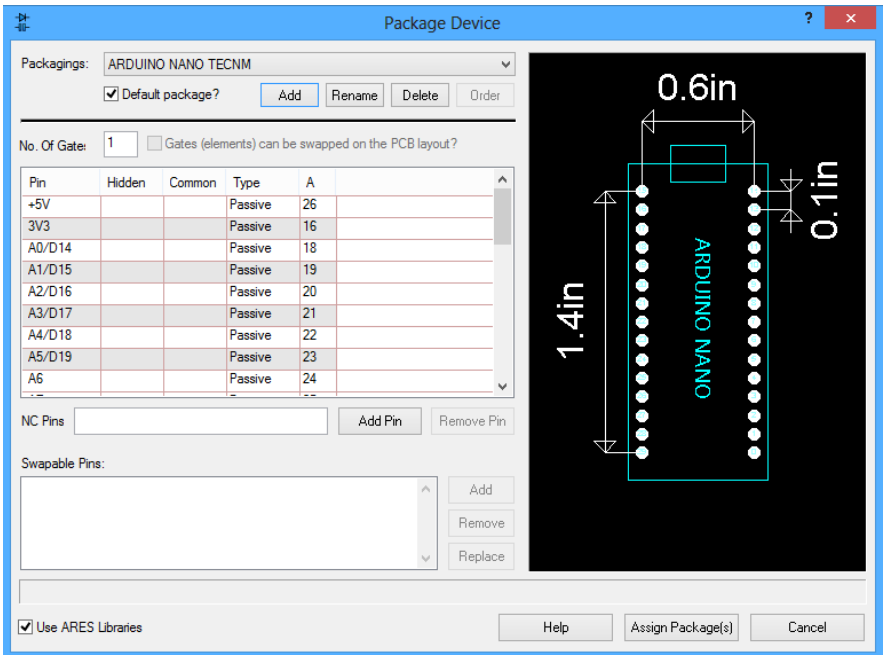


Imagen 65. Asignación del dispositivo.

Una vez terminada la asignación se procede a dar click en “Assing Package” > Next >Next > Next y por último se selecciona la categoría del dispositivo y en su defecto también una breve descripción del mismo para finalizar dando OK. Realizando este proceso se tendría por completo el dispositivo, listo para ser utilizado.

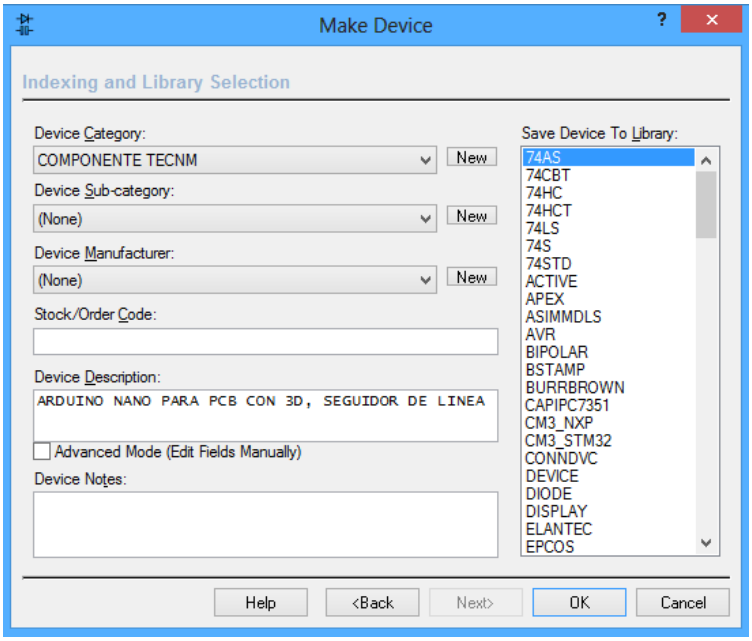


Imagen 66. Categoría del dispositivo.

Para obtener el nuevo dispositivo realizado se tiene que eliminar el Arduino Nano con el que ese inicio el trabajando, para esto se tiene que seleccionar en la de herramientas el dispositivo de Arduino Nano realizado y colocarlo en el espacio de trabajo de Proteus. Es muy importante mencionar que todos estos dispositivos realizados son únicamente para la elaboración de la PCB y no son útiles para realizar simulaciones.

El objetivo principal de realizar estos dispositivos es debido a las conexiones internas que la placa posee, por tal motivo una vez concluidos todos los dispositivos realizados, se procederá a realizar las conexiones correspondientes.

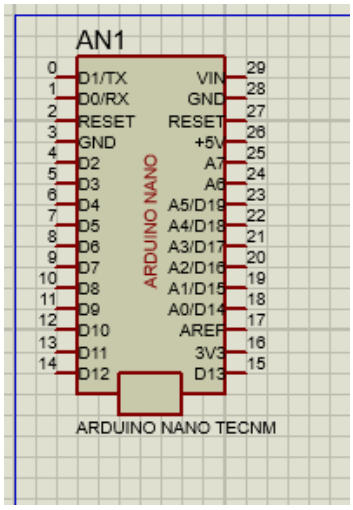


Imagen 67. Dispositivo Arduino Nano.

4.3.2 Puente H TB6612FNG

Realizar el TB6612FNG utiliza el mismo procedimiento que el Arduino Nano, partiendo de realizar un recuadro en el espacio de trabajo se deben seleccionar los pines cortos y colocar 8 por lado, recordando que las “x” deben quedar contrarias al recuadro del dispositivo. Para cuestiones estéticas, y de referencia se deben colocar tres recuadros más dentro del recuadro principal, y a uno de ellos se le colocará el nombre del dispositivo.

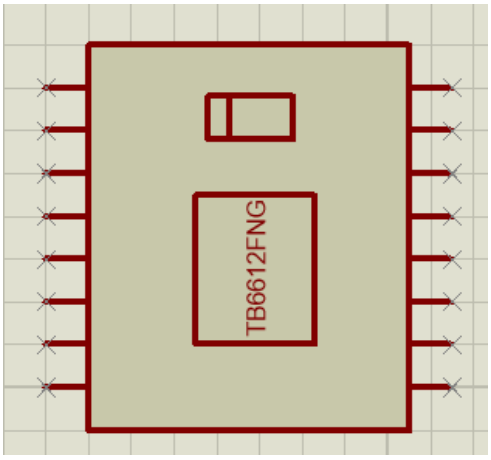


Imagen 68. Recuadro para TB6612FNG.

Una vez terminado el recuadro con los pines, se tiene colocar el nombre correspondiente a cada pin, iniciando de izquierda a derecha, en forma de U con el mismo orden del empaquetado. La siguiente imagen es una referencia de los nombres utilizados para los pines, es importante respetar el orden para que al momento de soldar los pines al componente físico no exista ningún problema o se tenga que recurrir a soldar el dispositivo de forma distinta. La finalidad de seguir el orden correcto es evitar en la medida de lo posible el desperdicio de material, o la falla de los componentes, los cuales a pesar de no ser todos caros siguen representando una inversión económica.

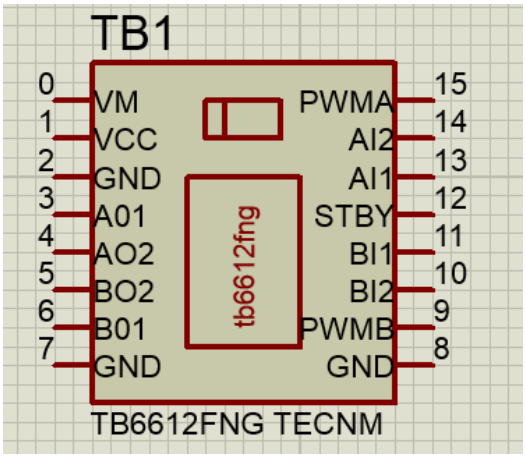


Imagen 69. Orden de pines TB6612FNG.

Una vez terminado el llenado de nombres se selecciona todo el dispositivo y se da inicio al proceso del empaquetado seleccionando la opción “Make Device”, llenando toda la información solicitada tal cual se hizo con el Arduino Nano pero ahora para el TB6612FNG, siguiendo el mismo proceso. Se debe prestar mayor atención al recuadro que añade al empaquetado del TB6612FNG por el tema de Pads, donde todos deben estar en blanco y no rosa, tal como se muestra a continuación.

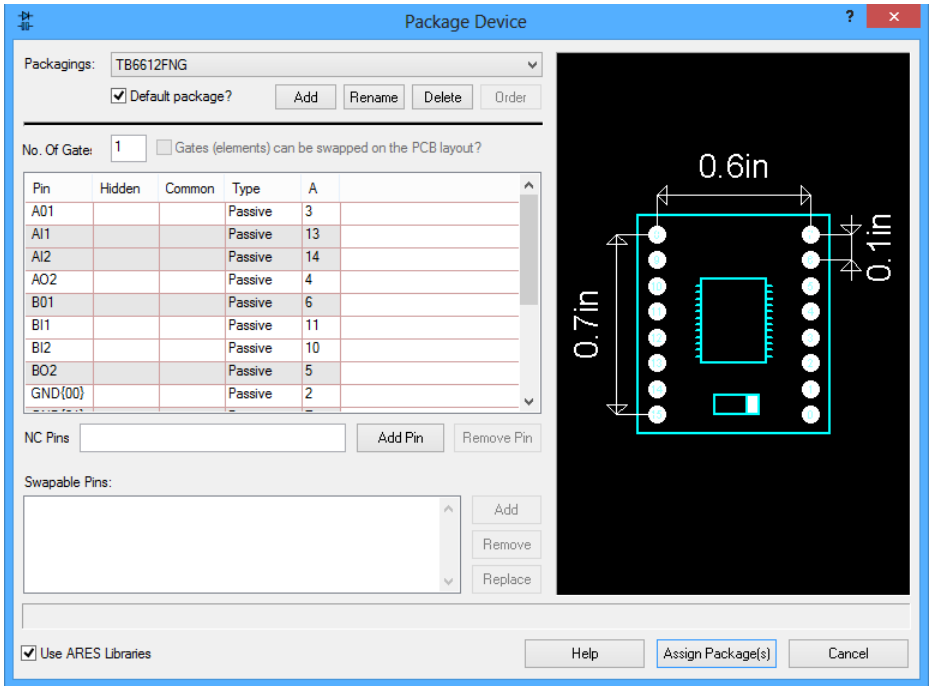


Imagen 70. Añadiendo Empaquetado TB6612FNG.

Terminada la asignación se debe dar click en “Assing Package” > Next >Next > Next y por último seleccionar la categoría del dispositivo con una breve descripción del mismo para finalizar dando click en OK. Realizando este proceso el dispositivo está completo y listo para ser utilizado. Lo último a realizar es eliminar el TB6612FNG con el que se inició el trabajo, seleccionando en la parte de herramientas el dispositivo realizado colocándolo en el espacio de trabajo de Proteus.

4.3.3 Pulsador

Para realizar el pulsador se inicia seleccionando en la barra de herramientas la opción “Pick Device” (P azul ubicada por encima de las opciones para los componentes utilizados), posteriormente en la ventana de búsqueda de componentes, se debe escribir la palabra “BUTTON” y seleccionar la primer opción, para después dar click en aceptar y colocar el componente en el espacio de trabajo.

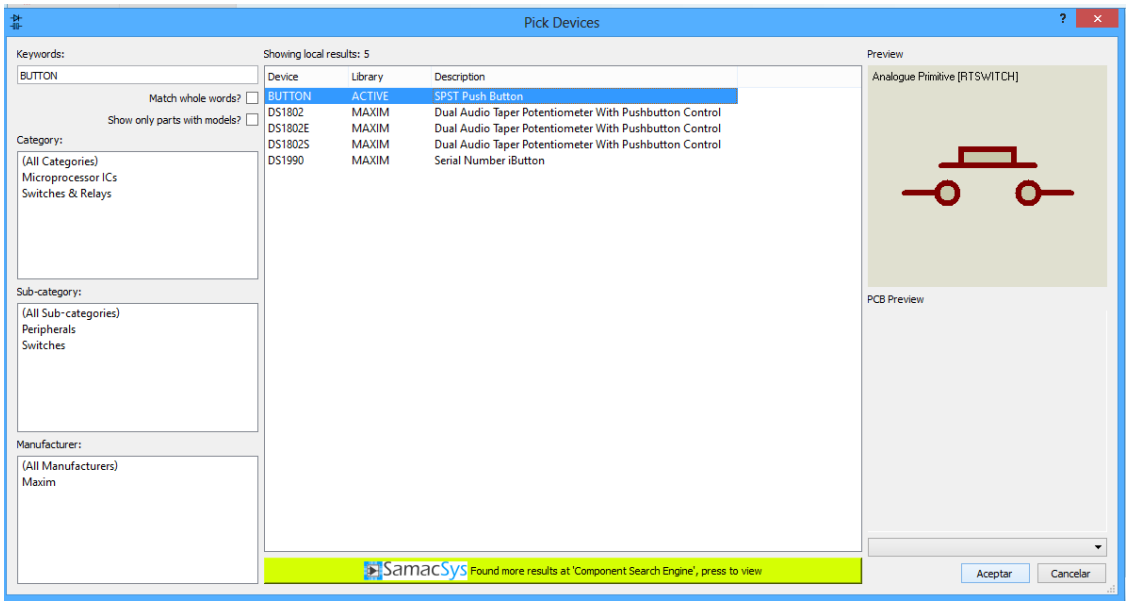


Imagen 71. Selección de botón.

Lo siguiente es descomponer el dispositivo, para ello se debe seleccionar el botón, dar click derecho sobre él y seleccionar la opción “Descompose” que desplegara el contenido del componente en la parte inferior, una vez la información sea visible se debe seleccionar y eliminar.

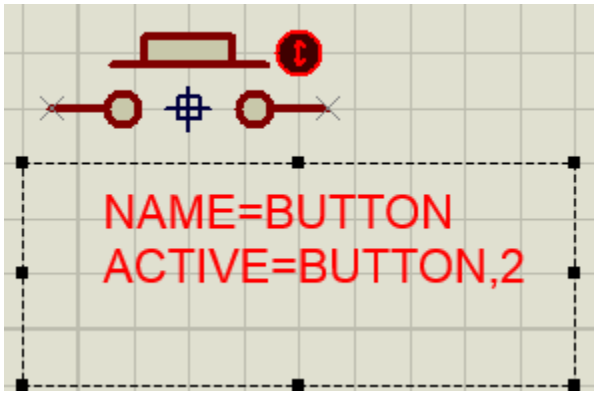


Imagen 72. Descomponiendo botón.

Después de eliminar la información del botón se debe seleccionar la opción “Make Device” para realizar el llenado de información del dispositivo tal cual se ha realizado con los dos dispositivos anteriores, tomando mayor importancia a añadir el empaquetado del dispositivo.

Una vez en la pestaña para añadir el empaquetado, se debe dar click en “Add” para buscarlo y agregarlo, escribiendo en el buscador el nombre con el cual se registró al pulsador para después agregarlo, cabe mencionar que los Pads aparecerán en color rosa, lo cual indica que no hay ningún problema con ellos. Lo único que resta por hacer es enumerar los Pads de la siguiente manera: 1 Arriba y 0 Abajo y con ello los Pads cambiarán a color blanco.

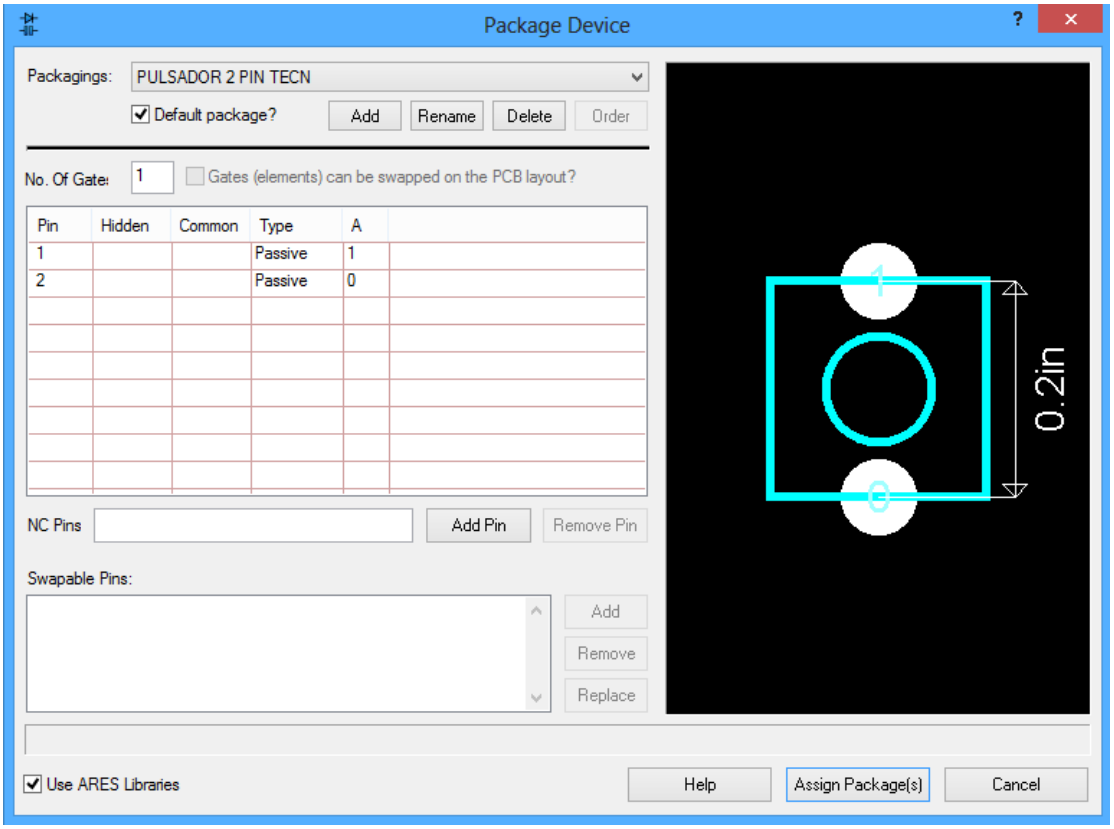


Imagen 73. Colocando numeración a pads.

Una vez terminada la asignación se debe dar click en “Assing Package” > Next >Next > Next para seleccionar la categoría del dispositivo con una breve descripción del mismo y finalizar dando click en OK. Realizando este proceso el dispositivo estará completo y listo para ser utilizado. Lo único que resta es eliminar el pulsador con el que se inició el trabajando, después se debe seleccionar en la parte de herramientas el dispositivo realizado y colocarlo en el espacio de trabajo.

4.3.4 Espadines

Para realizar los espadines de 11 y de 3 el proceso es el mismo, primero en el buscador de componentes de Proteus (P en color azul) en la ventana de búsqueda, se debe escribir “Block” para desplegar todos los tipos de espadines con los que cuenta Proteus. Para realizar el espadín de 3 pines se puede utilizar el espadín de 2 o de 4 y para el espadín de 11 se puede utilizar el espadín 10 pines.

4.3.4.1 Espadín de 11 pines

Para realizar el espadín de 11 pines se debe seleccionar el componente “CONN-H10” y dar click en aceptar para colocarlo sobre el espacio de trabajo.

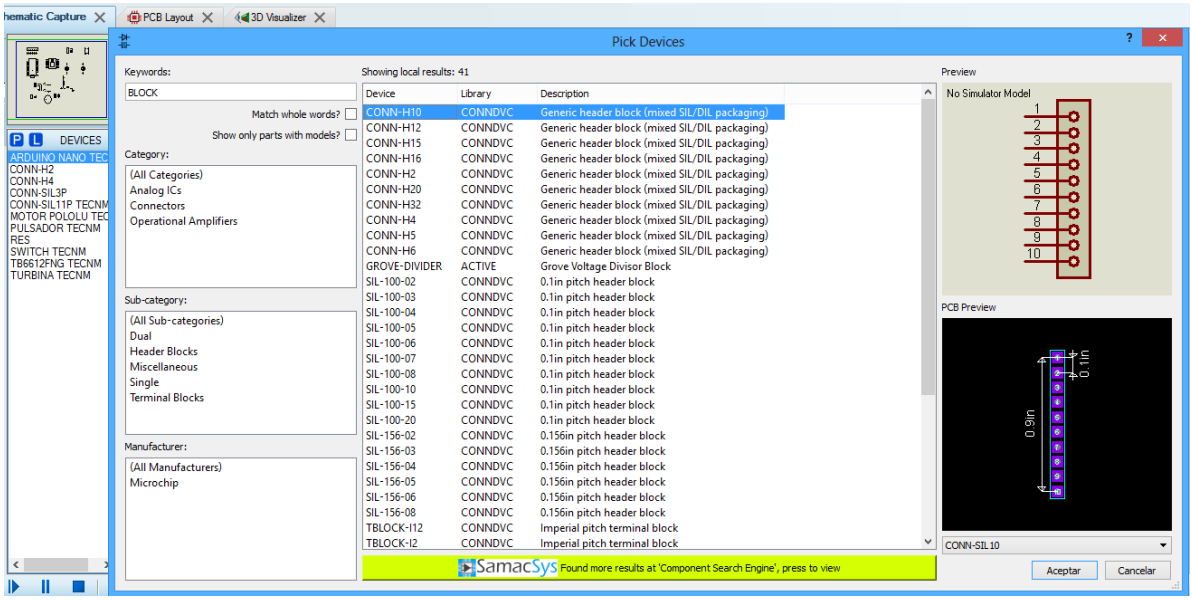


Imagen 74. Selección de espadín de 10.

Una vez colocado en el espacio de trabajo, se selecciona al componte para aplicar la opción “Descompose” y descomponer el espadín de 10 para eliminar todas sus especificaciones. Después el componente podrá ser editado. Para realizar la edición del componente, se debe expandir el área

de los espadines hacia abajo, seleccionando todos los pines para recorrerlos, copiar uno de ellos y colocarlo en la parte superior del recuadro que encapsula a los demás pines, después se debe cambiar el número del nuevo pin por el número 0, para que los pines estén enumerados del 0 al 10 teniendo como resultado 11 pines en total para el espadín.

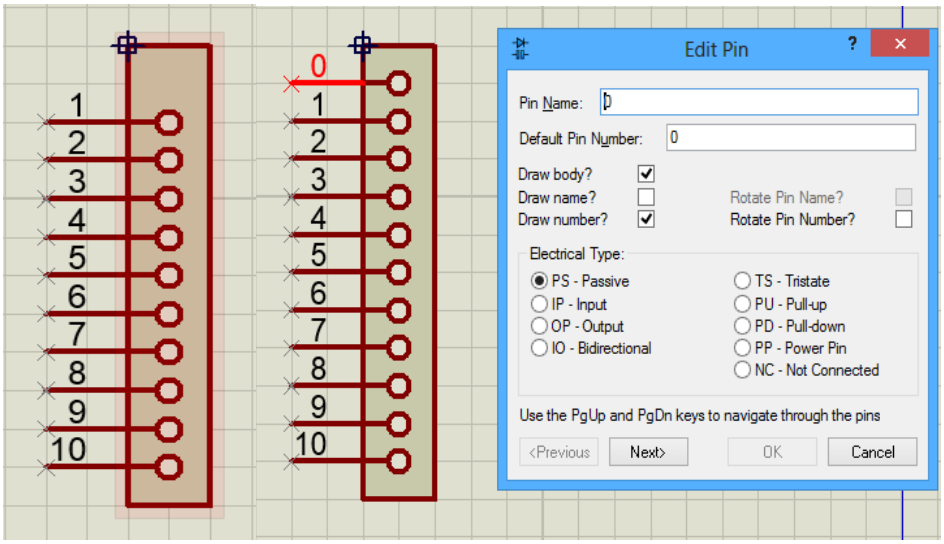


Imagen 75. Cambiando de Espadín de 10 a 11.

Lo que resta por hacer es realizar el llenado de datos del dispositivo, seleccionadolo y dando click derecho para utilizar la opcion “Make Device”, para el primer recuadro se debe colocar la informacion solicitada, siendo hasta el recuadro packagingns donde debe prestarse mayor atención, puesto que ahí se seleccionara el empaquetado del espadin de 11 realizado anteriormente. Lo unico que resta por hacer es verificar que todos los recuadros del espadín esten en blanco, una vez confirmado esto se puede proceder a dar click en “Assign Package” > Next >Next > Next y por último seleccionar la categoría de nuestro dispositivo junto a una breve descripción del mismo para finalizar dando click en OK.

Realizado este proceso el dispositivo estará completo y listo para ser utilizado. Lo único restante es eliminar el espadín con el que se inició el trabajo, para después seleccionar en la parte

de herramientas el dispositivo espadín de 11 realizado y colocarlo en el espacio de trabajo de Proteus.

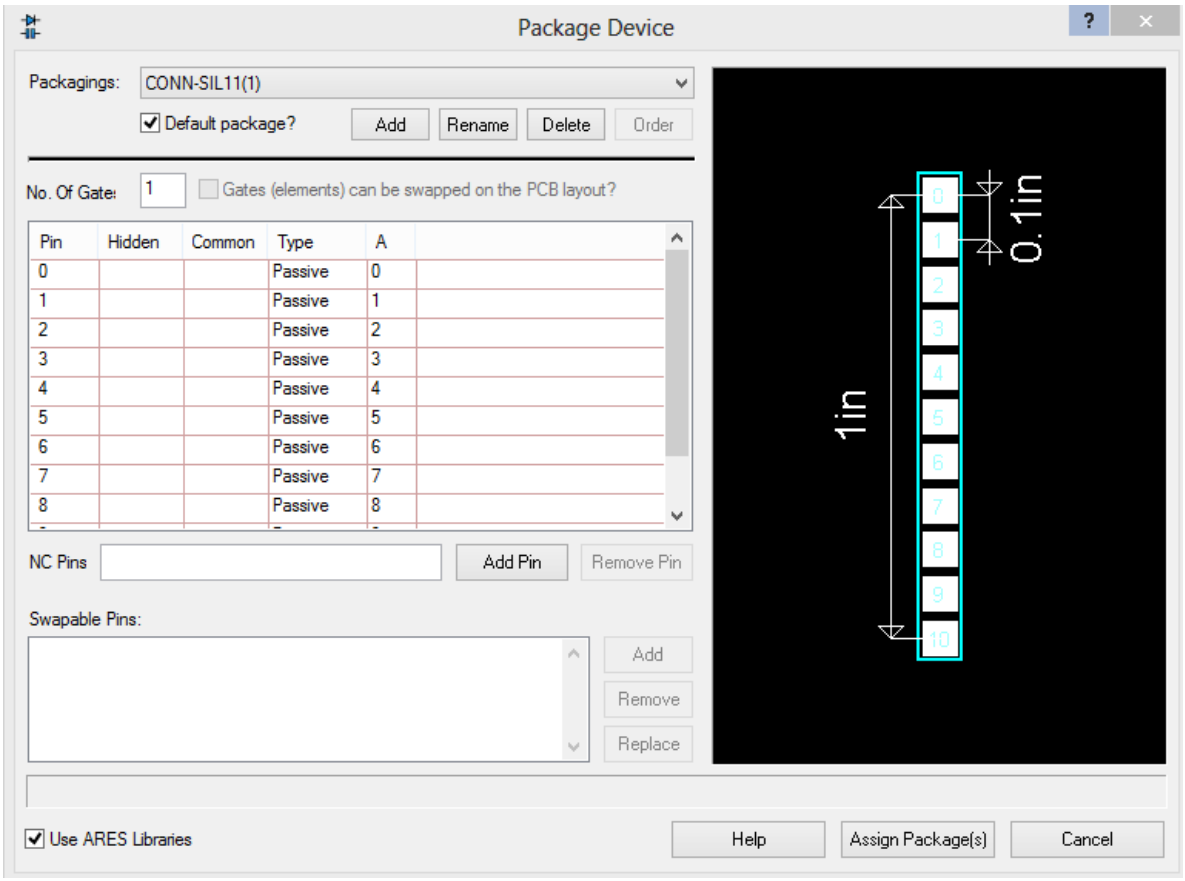


Imagen 76. Verificación de empaquetado.

4.3.4.2 Espadín de 3

Para realizar el espadín de 3 pines se tiene que seleccionar el espadín de 4 pines colocando en el buscador de componentes CONN-SIL4, se tiene que descomponer el dispositivo y eliminar uno de los pines que están dentro del encapsulado, posterior a ello se deben reenumerar los pines iniciando de 0 a 2 o en su defecto dejarlo tal cual. También existe la posibilidad de que Proteus ya cuente con dicho dispositivo y por tanto únicamente se tendría que descomponer, eliminar sus datos y realizar el proceso de empaquetado para agregar el empaquetado realizado. Lo único que resta es

verificar que los Pads de perforación estén en color blanco y no rosa para evitar cualquier error, posteriormente se selecciona “Assign Package” > Next >Next > Next y por último se hace la selección de la categoría de dispositivo junto a una breve descripción del mismo para finalizar dando click en OK. Terminado el proceso del empaquetado se debe eliminar el espadín con el que se inició el trabajo, seleccionar en la parte de herramientas el dispositivo espadín de 3 realizado y colocarlo en el espacio de trabajo de Proteus.

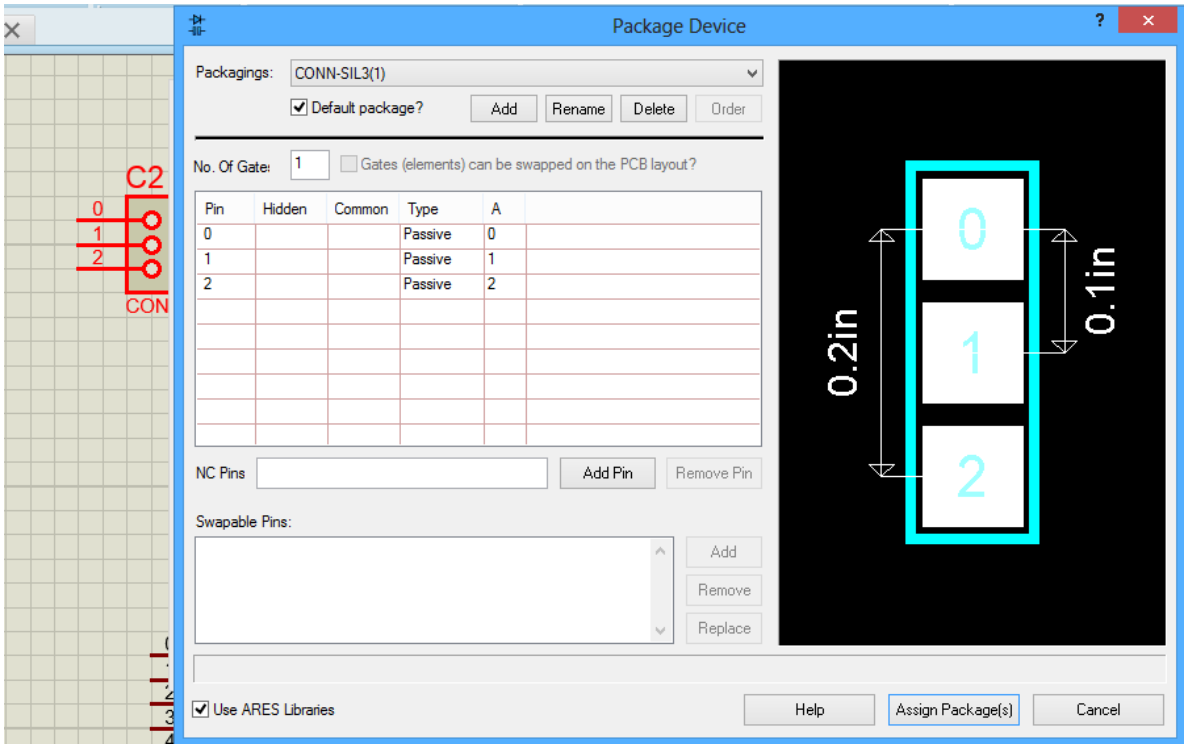


Imagen 77. Verificación de empaquetado.

4.3.5 Microswitch

Para realizar el dispositivo Microswitch serán utilizados dispositivos que ya vienen incluidos en Proteus, para esto en la barra de búsqueda se tiene que escribir la palabra “SWITCH” eligiendo el Switch con el código “SW-SPDT-MOM” o en caso de no tenerlo, puede ser utilizado el código “SW-SPDT”. Después de seleccionar el Switch a utilizar se tiene que descomponer el dispositivo

para poder eliminar la información con la que cuenta y así poder trabajar con él. Cuando la información es eliminada se debe dar click derecho al dispositivo para iniciar con el proceso de empaquetado utilizando la opción “Make Device”. Este proceso no presenta gran complejidad puesto que es el mismo que se ha realizado para todos los dispositivos, entonces prestando especial atención a la ventada donde se asigna el empaquetado del dispositivo, se debe agregar en “Add” el empaquetado realizado con anterioridad, colocando el nombre del empaquetado con el que se registró. Lo único pendiente por revisar es que los Pads de perforación estén colocados de manera correcta para después terminar el proceso dando click en “Assign Package” > Next >Next > Next para seleccionar la categoría del dispositivo y también una breve descripción del mismo finalizando con un click en OK. Lo último a realizar es eliminar el Switch con el que se inició el proceso, seleccionar en la parte de herramientas el dispositivo Switch realizado y colocarlo en el espacio de trabajo de Proteus.

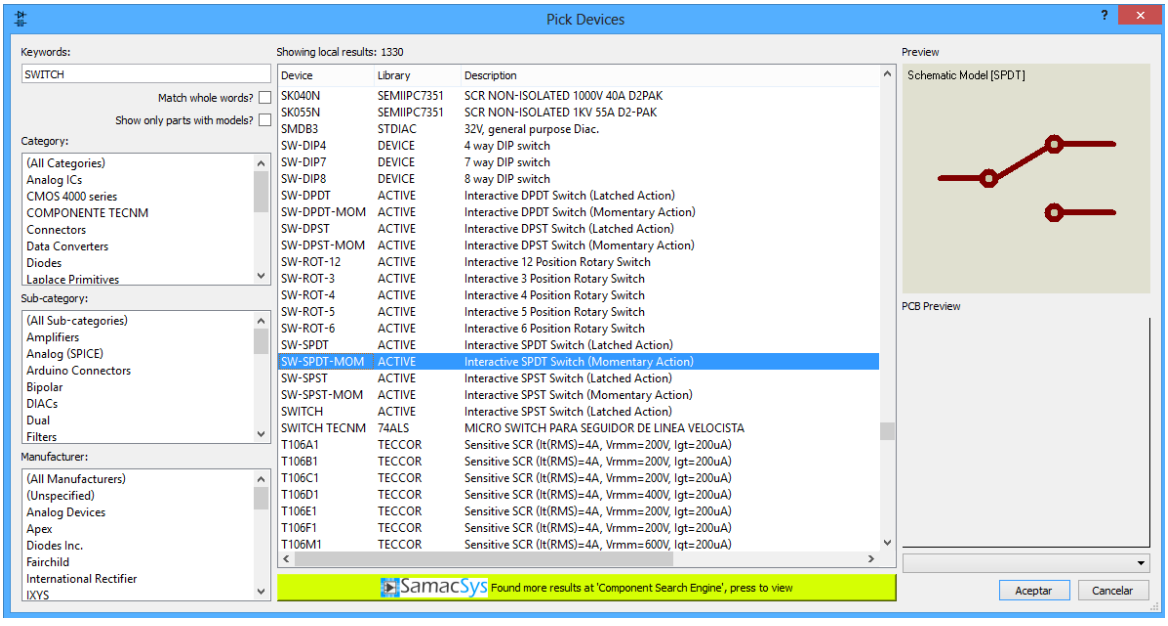


Imagen 78. Búsqueda del Switch.

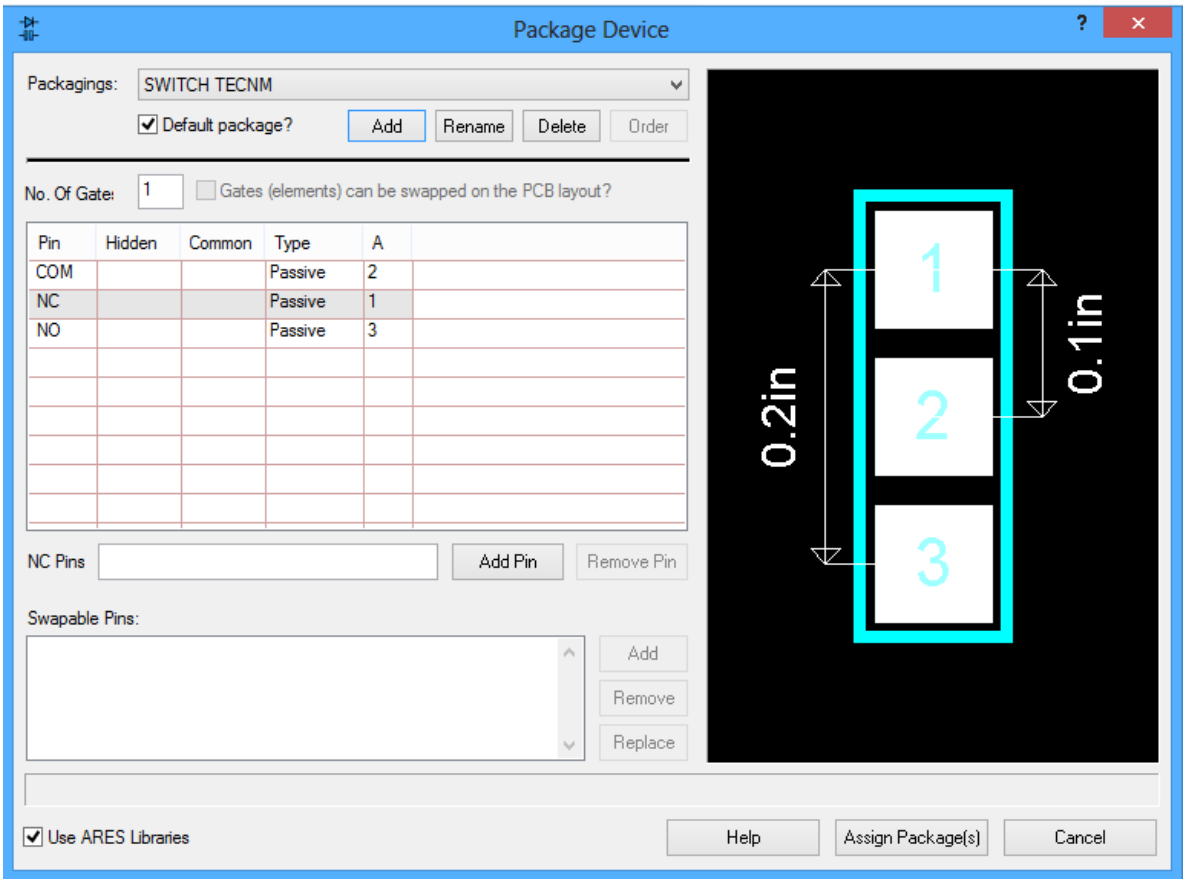


Imagen 79. Cargando empaquetado de Switch.

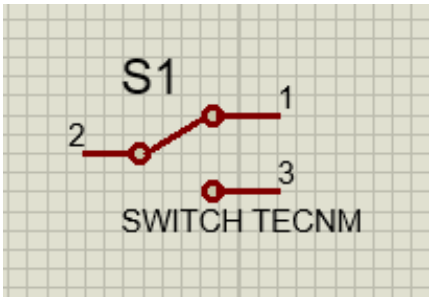


Imagen 80. Dispositivo Switch.

4.3.6 Motor

Para poder trabajar con el motor, lo primero que se debe realizar es la búsqueda de un motor normal de DC en el buscador de dispositivos de Proteus. Una vez localizado debe ser insertado en

el espacio de trabajo de Proteus para poder descomponerlo. Para referenciar el motor se pueden seguir las especificaciones del mismo que indican lo siguiente: “Simple DC Motor Model”.

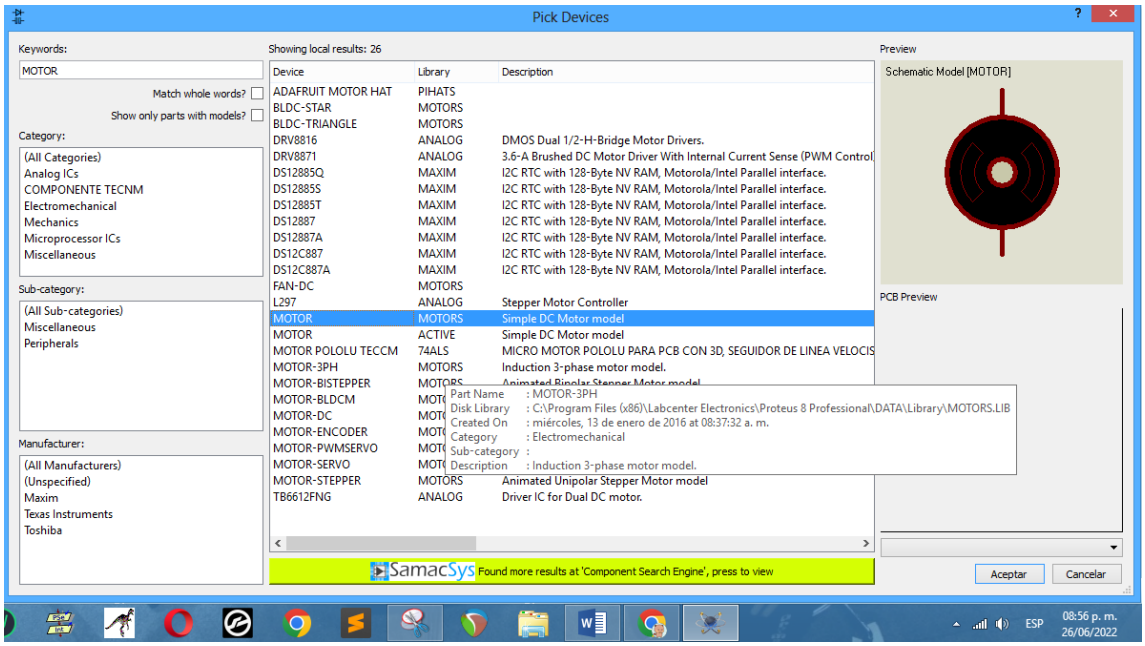


Imagen 81. Búsqueda de motor DC normal.

Con click derecho procedemos a descomponer el dispositivo, después se tiene que eliminar la informacion que contiene el dispositivo para en base a ese motor poder realizar los motores tipo Pololu para Proteus.

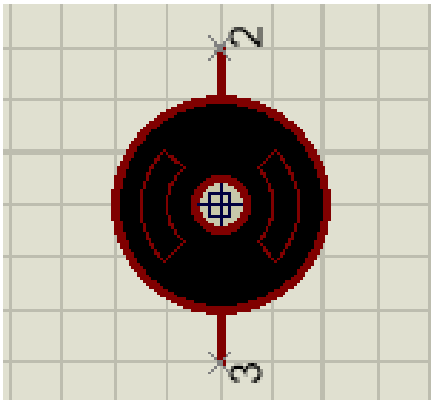


Imagen 82. Información de motor.

Se debe asignar la numeración a los pines del motor, dando el número 2 al pin superior y número 3 al pin inferior. Para el llenado de datos se debe dar click derecho sobre el motor, después utilizando la opción “Make Device” se podrá comenzar con el llenado de datos que solicita Proteus, para el segundo recuadro se debe dar click en la opción “Add/Edit”, y posteriormente en el siguiente recuadro en la parte superior se debe dar click en “Add” para localizar en el buscador el empaquetado del motor tipo Pololu. Como nota, para este caso en particular solo el pin 0 y 1 deben aparecer en color rosa, el pin 2 y 3 en color blanco, puesto que esos dos pines serán las conexiones entre los motores y la placa.

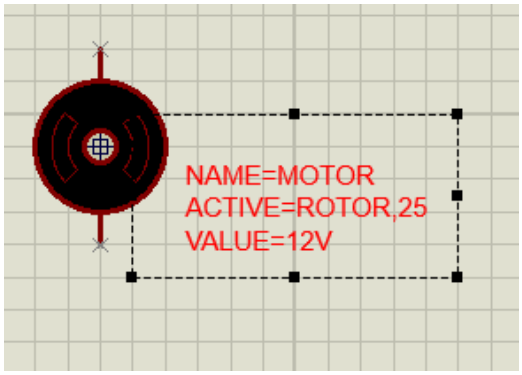


Imagen 83. Numeración de pines.

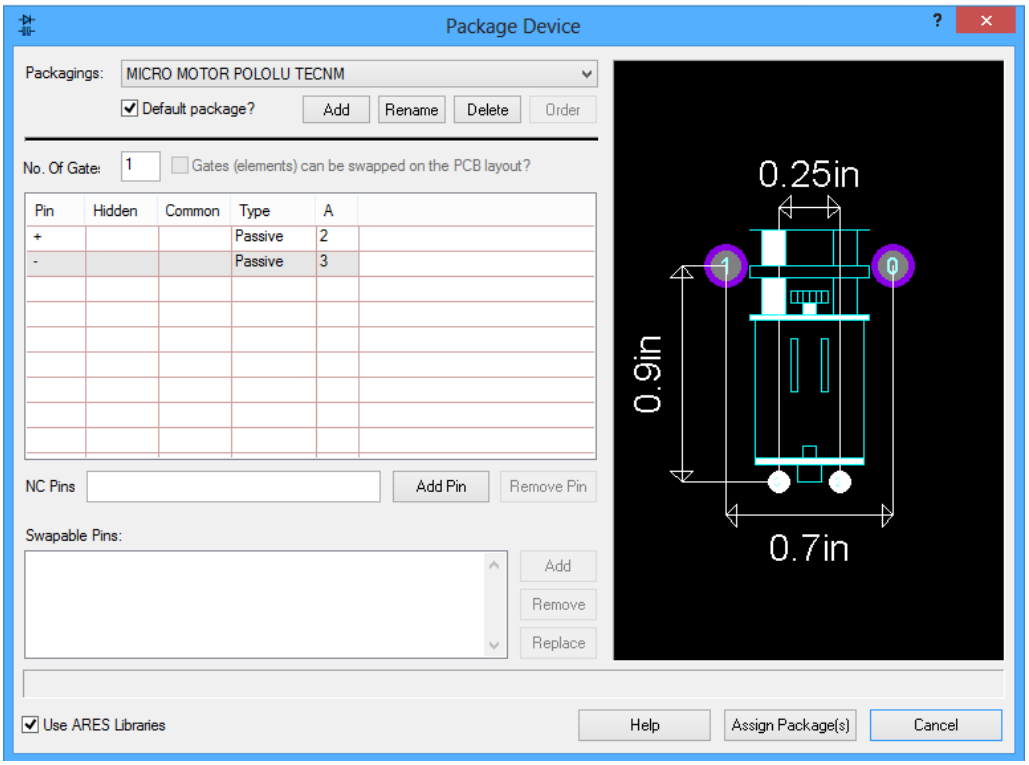


Imagen 84. Añadiendo el empaquetado del motor.

Para terminar el proceso se debe dar click en “Assign Package” > Next >Next > Next y por último seleccionar la categoría del dispositivo junto a una breve descripción del mismo para finalizar dando click en OK. Lo último a realizar es eliminar el motor del principio, seleccionar en la parte de herramientas el Motor realizado y colocarlo en el espacio de trabajo de Proteus.

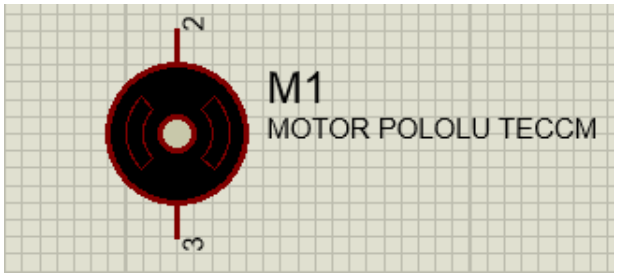


Imagen 85. Dispositivo Motor Tipo Pololu.

4.3.7 Turbina

Para realizar la turbina el proceso es bastante sencillo, lo único que se debe hacer es crear un círculo de una medida que se considere adecuada con ayuda de la herramienta para figura circular, no siendo tan pequeña pero tampoco llegando a ser tan exageradamente grande, y colocarla en el espacio de trabajo. Después se le puede colocar el nombre de "Turbina" al dispositivo en la parte central. Adicional al nombre se debe colocar un pin a la turbina para poder crear el dispositivo, entonces con ayuda de la herramienta "Device Pins Mode" y los pines "Short" deberá ser colocado un pin, al cual con doble click sobre él podrán ser editadas sus propiedades desmarcando las casillas Draw Body, Draw number y Draw Name e inmediato a ello se colocara un nombre cualquiera para Pin Name y Default Pin Name, por ejemplo: R. Quedando de la siguiente manera:

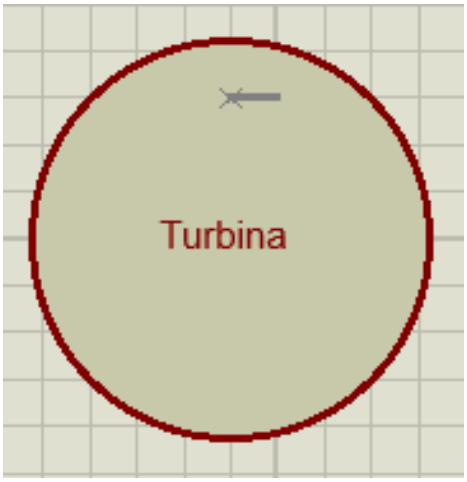


Imagen 86. Diseño de turbina.

Después se tiene que iniciar el proceso del dispositivo para la turbina, dando click derecho sobre ella se puede comenzar con el llenado de información, centrando principalmente atención a la ventana que solicita el empaquetado realizado con anterioridad de la turbina, una vez agregado el empaquetado de la turbina lo único que queda es asignar un 1 al pin de la turbina para que todo

funcione en orden. Para terminar el proceso se debe dar click en “Assign Package” > Next >Next > Next para por último seleccionar la categoría del dispositivo y una breve descripción del mismo finalizando con click en OK.

Lo último a realizar es eliminar la turbina con la que se trabajó, seleccionar en la parte izquierda de herramientas el dispositivo Turbina realizado y colocarlo en el espacio de trabajo de Proteus.

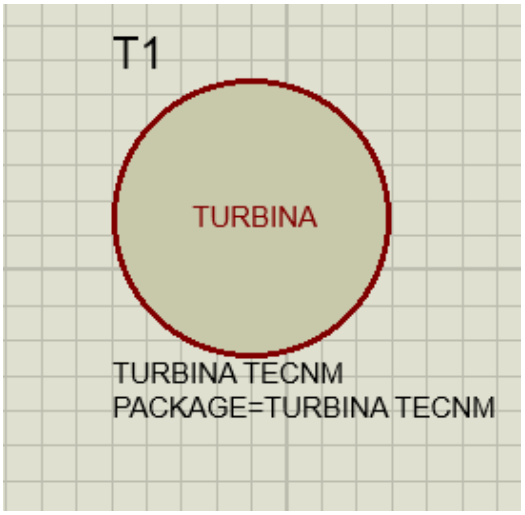


Imagen 87. Dispositivo de turbina.

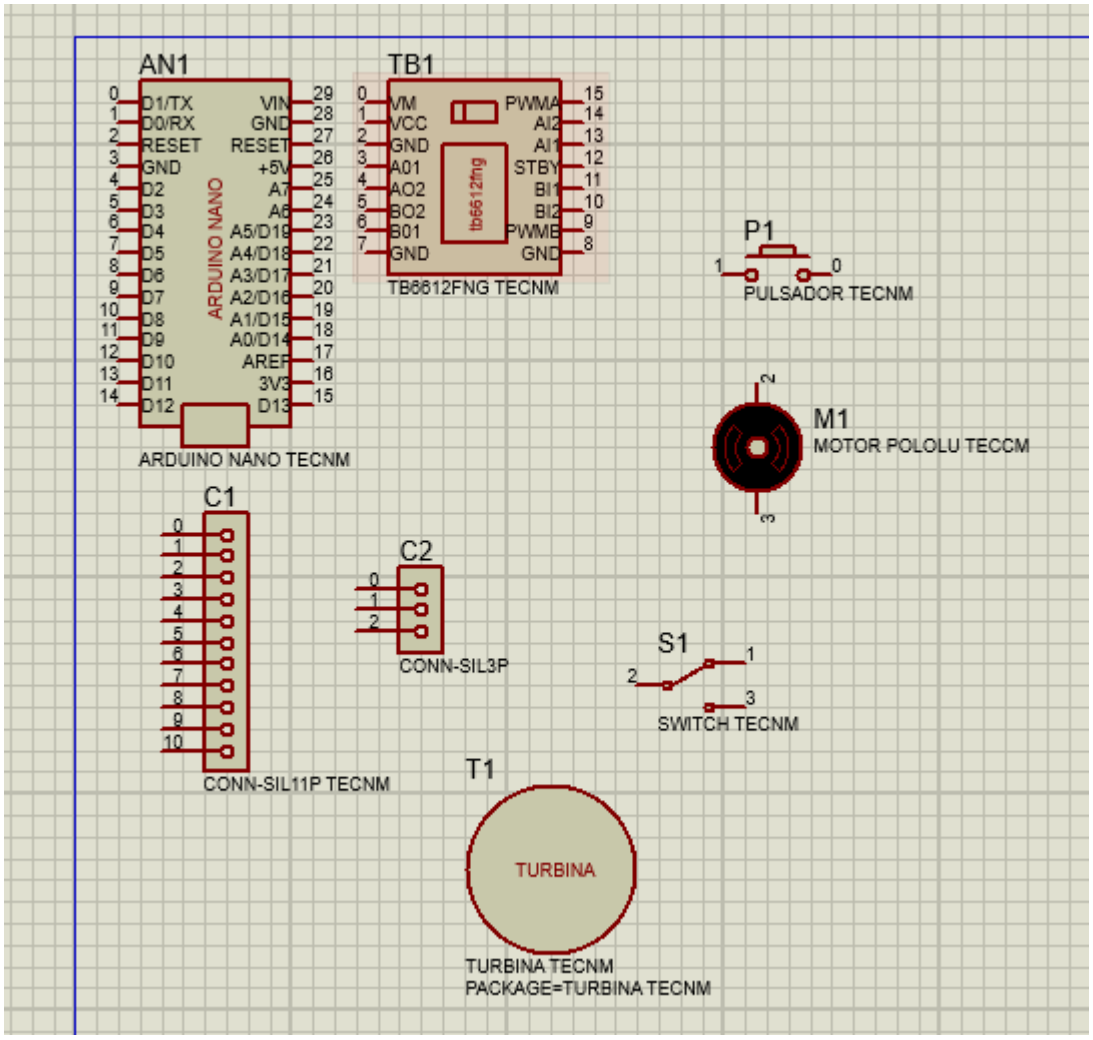


Imagen 88. Todos los dispositivos.

En la imagen anterior se pueden observar todos los dispositivos realizados, colocados sobre el espacio de trabajo de Proteus. En caso de no haber eliminado ningún dispositivo con los que se realizaron los nuevos, estos podrán ser eliminados sin ningún problema, puesto que en la parte izquierda de Proteus se deben visualizar los nombres de los nuevos dispositivos realizados, después lo único que se debe hacer es seleccionar uno por uno y enviarlo al espacio de trabajo tal cual se muestra en la imagen anterior. Una característica que tienen estos dispositivos y por el cual se puede verificar que el trabajo realizado ha sido correcto es por las siglas “TECNM”.

Una vez colocados todos los dispositivos en el espacio de trabajo, cambiando de la ventana "Schematic Capture" a la ventana de "PCB Layout" en la opción "Component Mode" podrán observarse todos los empaquetados construidos, de ahí pueden ser colocados todos los empaquetados realizados con el orden de ejemplo siguiente, y lo único que se debe observar es que no existan conexiones entre los pines de un mismo dispositivo, a excepción del Arduino Nano, que tiene en dos pines distintos los mismos nombres. En caso de haber una conexión errónea en los demás dispositivos se tendría que revisar que los nombres de sus pines no sean el mismo; después de corregir el error de nombres en los pines se tiene que corregir en caso de ser necesario la información que viene debajo de tal dispositivo, para después realizar el proceso de dispositivo, solo dando "Next" a todo puesto que la información agregada se guarda por defecto en la información del dispositivo.

Para el caso del Arduino Nano los pines que presentan conexión son: 2 con 27 y 3 con 28.

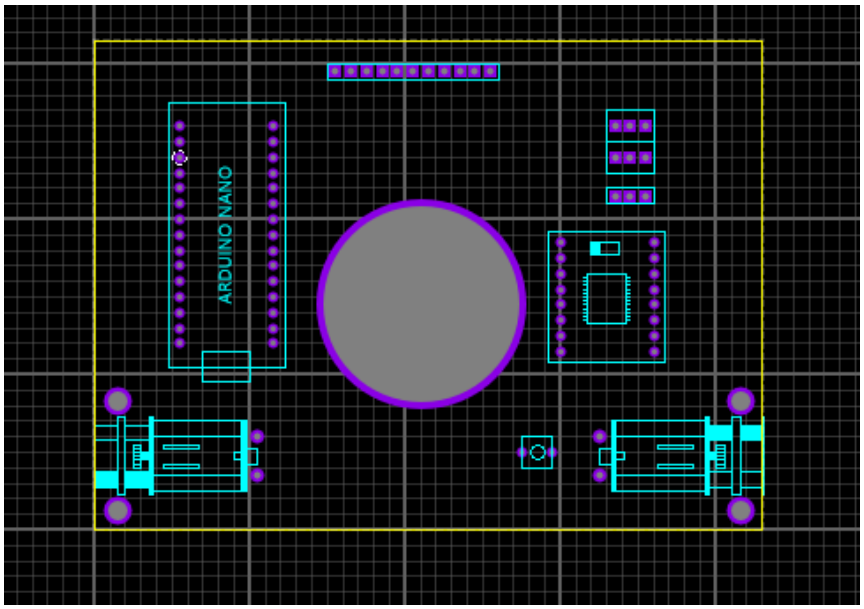


Imagen 89. Placa con dispositivos.

4.4 Esquema de conexiones entre los dispositivos

Para realizar las conexiones de los componentes, lo primero a realizar, es la distribución de los dispositivos en el espacio de trabajo de Proteus, (se puede tomar de ejemplo la siguiente imagen) para ello serán utilizados, el Arduino Nano, el puente H TB6612FNG, un espadín de 3 y de 11 pines, dos Motores, una Turbina, conexión a VCC y conexión ground (tierra), un Switch y un Pulsador.

Se debe utilizar la herramienta “Terminals Mode” para realizar las conexiones entre dispositivos sin tener que utilizar las conexiones normales, las cuales pueden llegar a ser un poco molestas en cuanto a maniobrabilidad entre dispositivos por el espacio de trabajo. Entonces para seleccionar esta herramienta en la parte izquierda de la pantalla se encuentra la opción “Terminals Mode”, una vez seleccionada se debe elegir la tercera opción de arriba hacia abajo que hace alusión al nombre “OUTPUT”.

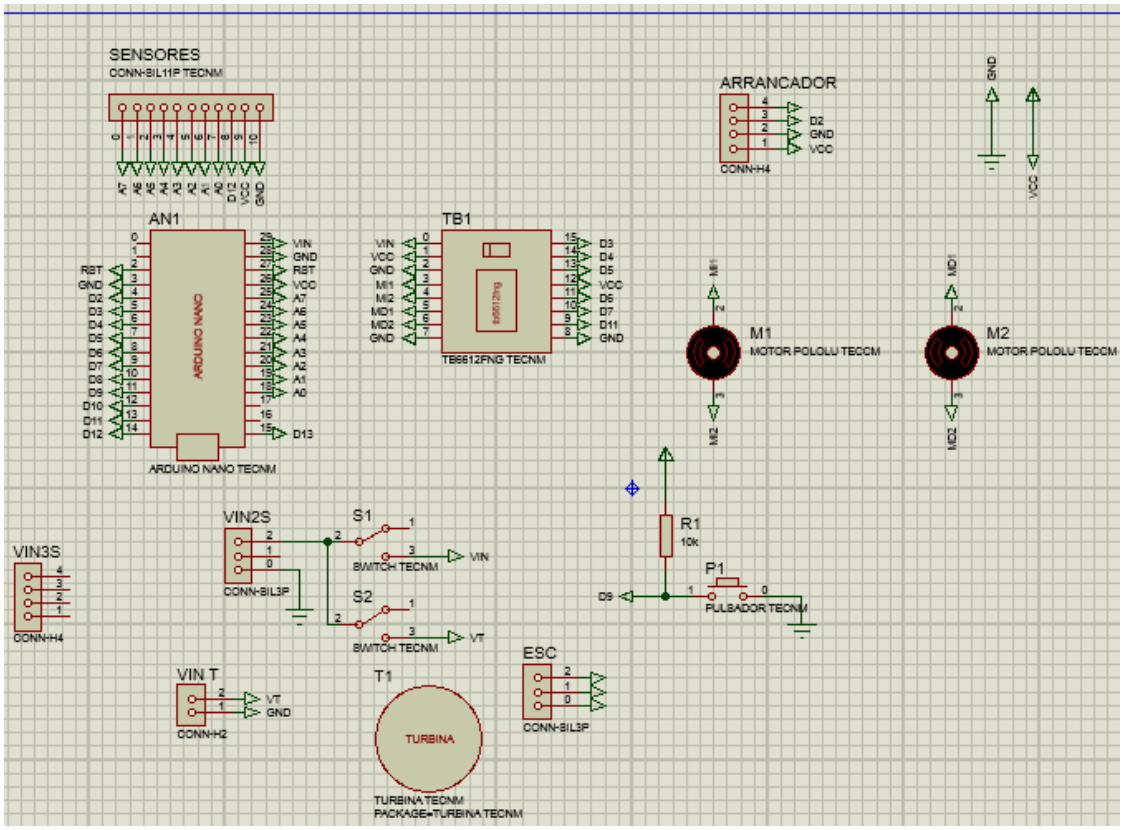


Imagen 90. Diagrama de conexiones entre dispositivos.

Para el Arduino Nano y el puente H TB6612FNG se colocan las terminales de salida casi justo al frente de cada uno de los pines de ambos dispositivos, para el Arduino Nano los pines 0, 1, 16 y 17 no llevarán conexión, por tanto esos pines quedaran libres de una terminal de salida. Para el TB6612FNG todos los pines llevaran un terminal de salida.

Justo después de colocar los terminales de salida se debe realizar la conexión entre los pines y los terminales de salida. En un principio no debería ser de preocupación que no aparezcan letras en las conexiones, puesto que aún no están definidas y el único enfoque por el momento es dar orden, distribución y terminales a todos los dispositivos que se van a utilizar.

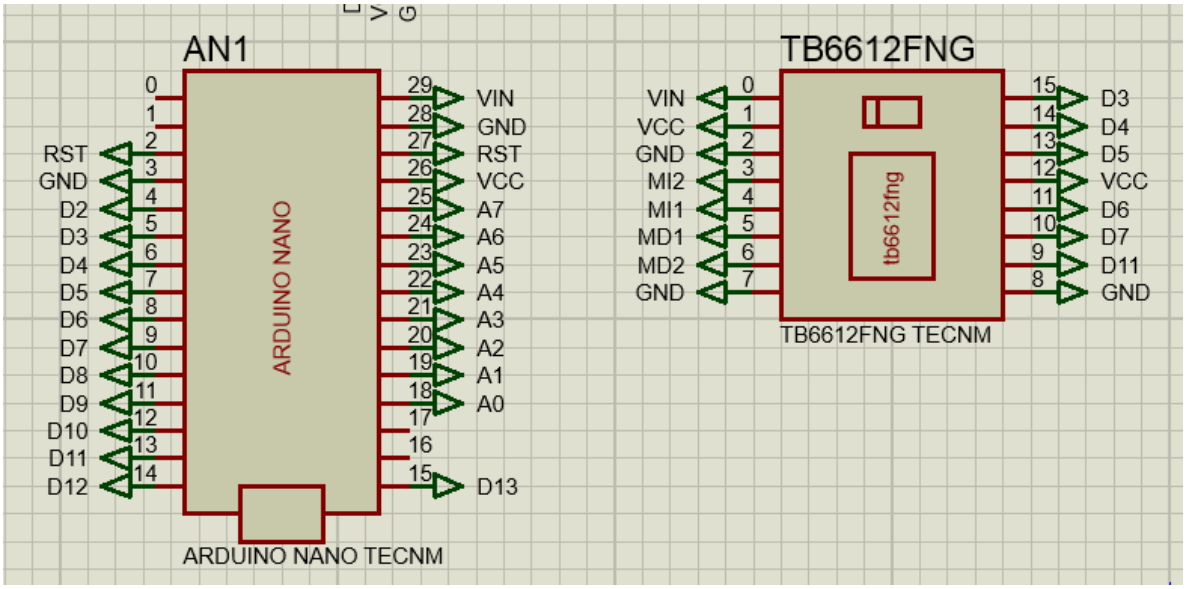


Imagen 91. Terminales de salida Arduino Nano y TB6612FNG.

En la parte superior al Arduino Nano se debe colocar el espadín de 11, y dando doble click sobre él se podrá modificar su nombre; para este caso se le designo el nombre de “Sensores” puesto que estos espadines serán las conexiones entre la placa y la barra multiplexada del seguidor de línea velocista, paralelo a ello apuntando hacia abajo se van a colocar las terminales de salida en cada uno de los pines del espadín de 11.

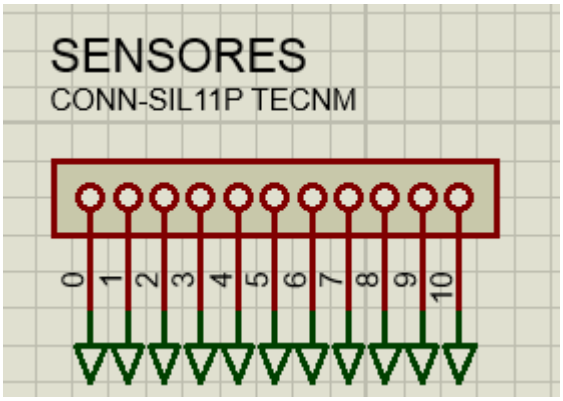


Imagen 92. Terminales para espadín de 11 (Barra de Sensores).

De forma continua pero al extremo contrario del espacio de trabajo se debe colocar un espadín de 4 y con doble click sobre él, cambiar su nombre por arrancador; siendo este el dispositivo que recibirá la señal del control (Juez) que activa al seguidor de línea, resulta ser muy importante para que funcione y pueda competir de manera profesional. Además, a un costado de este Arrancador se debe colocar una terminal de tierra y una terminal de VCC ubicadas en la barra de herramientas en el apartado “Terminals Mode” con los nombres “Ground” (Tierra) y “Power” (Energía), una vez colocados de forma individual se les debe colocar sus nombres y sus terminales de salida.

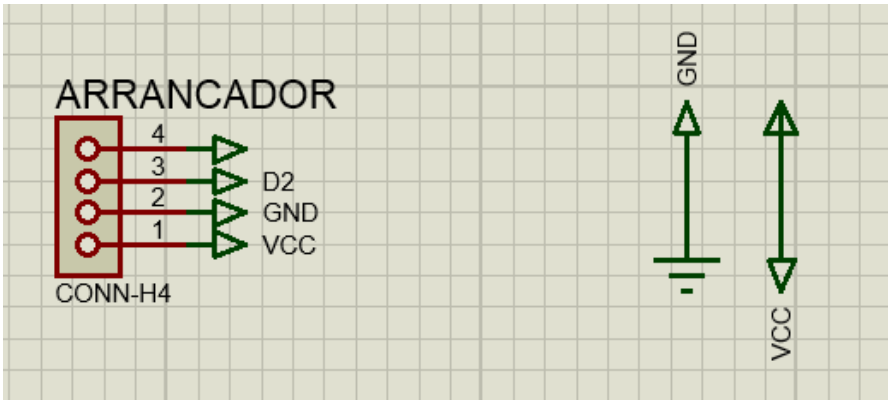


Imagen 93. Terminales para espadín de 4 (Arrancador) y tierra y VCC.

Debajo del arrancador, de la señal de tierra y VCC se deben colocar los dos motores a utilizar, y de igual forma se colocaran los terminales de salida en cada uno de los pines de los motores, siendo dos pines por motor el resultado será colocar 4 terminales de salida para ambos motores y la asignación de un nombre para cada motor, nombrándolos motor derecho y motor izquierdo.

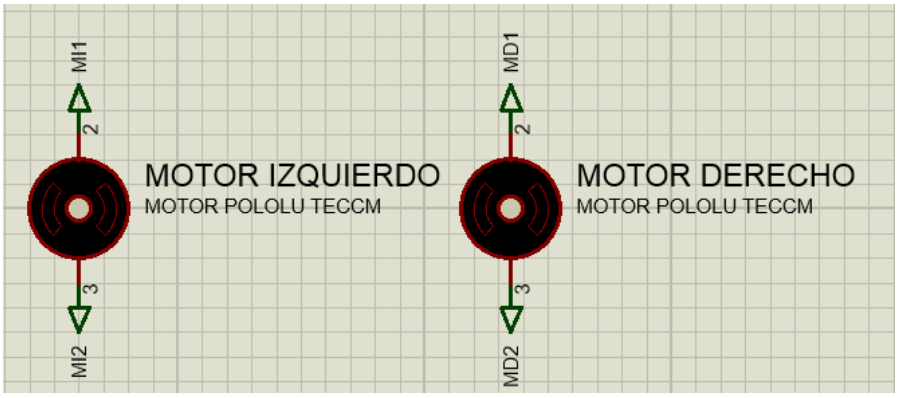


Imagen 94. Terminales para los motores.

Debajo de los motores se debe realizar una conexión entre un resistor de 10K, una conexión a VCC, a tierra y al Pulsador. Colocando en forma vertical la resistencia, en la parte superior se debe realizar una conexión a VCC, después en la conexión contraria se debe conectar el Pulsador a su pin izquierdo y en el pin derecho del pulsador a la conexión a tierra, justamente en la unión del resistor y del Pulsador se debe colocar el terminal de salida.

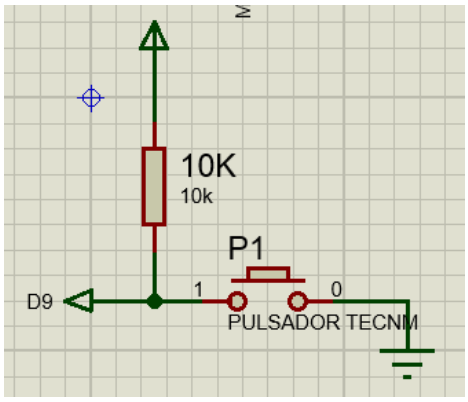


Imagen 95. Terminales para resistor y pulsador.

Otra conexión importante es la de un espadín de 3 a 2 Switch, de donde la salida número 2 del espadín tendrá una conexión para ambos Switch. El primer Switch deberá ir conectado del pin 2 a la salida 2 del espadín, y de su pin 3 se tendrá un terminal de salida, para el segundo Switch también deberá tener conexión de su pin 2 a la salida 2 del espadín con una terminal de salida para

el pin 3 del segundo Switch. Por último, el pin 0 del espadín debe tener una conexión a tierra. Es muy importante realizar las conexiones de los Switch debido a que el primero será el encargado de dar corriente a la mayoría de los componentes en la placa, mientras que el segundo se hará cargo de dar energía a la turbina, para que la turbina pueda realizar su función permitiendo crear agarre del seguidor sobre la pista.

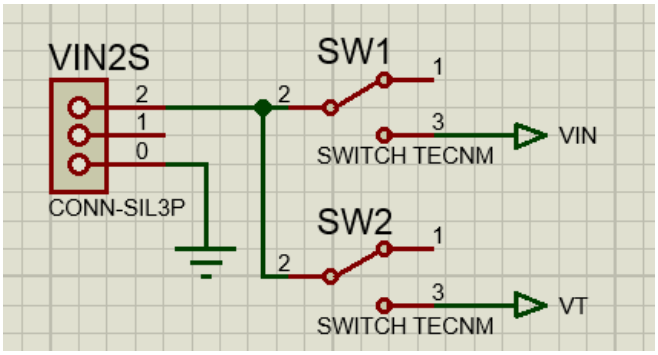


Imagen 96. Terminales para 2 Switch y un espadín de 3.

Lo único restante por hacer es colocar un espadín de 2, la turbina y un espadín de tres justo en ese orden de izquierda a derecha, debido a que así están las imágenes del proyecto del cual se está tomando por guía, aunque también se podría utilizar otro orden, lo único importante es que al momento de asignar las salidas de los dispositivos, se escriban de forma correcta y en el orden en que descrito. Entonces para el espadín de 2 se colocaran únicamente dos terminales de salida para cada uno de sus pines, para la turbina y para el espadín de 3 restante se colocaran para cada uno de sus pines una terminal de salida.

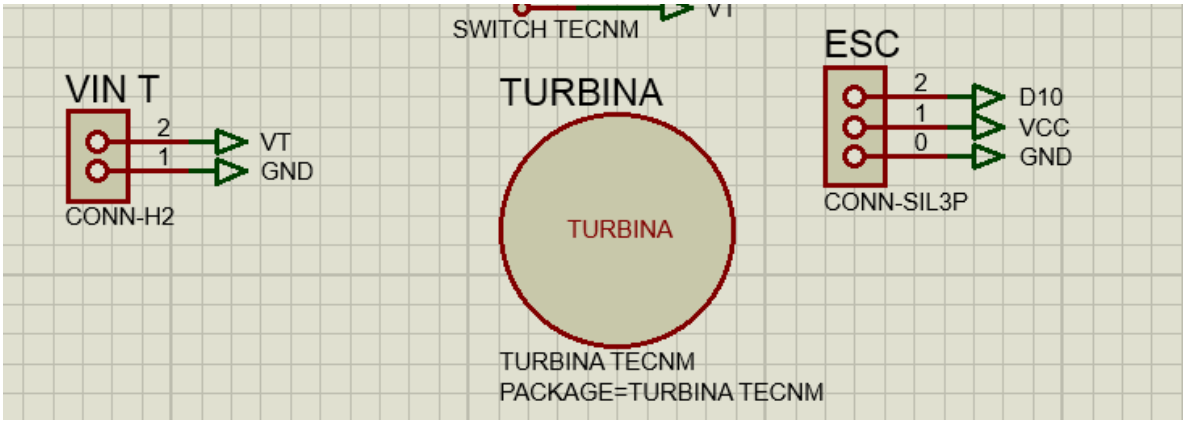


Imagen 97. Terminales para un espadín de 2, de 3 y la turbina.

4.4.1 Conexión de terminales para el arduino nano

Las conexiones para el Arduino Nano serán las siguientes en orden decente y en forma de lista comenzando por el pin 0, las “X” indicaran que ese pin no tiene conexión.

Pin 0 = X

Pin 1 = X

Pin 2 = RST

Pin 3 = GND

Pin 4 = D2

Pin 5 = D3

Pin 6 = D4

Pin 7 = D5

Pin 8 = D6

Pin 9 = D7

Pin 10 = D8

Pin 11 = D9

Pin 12 = D10

Pin 13 = D11

Pin 14 = D12

Pin 29 = VIN

Pin 28 = GND

Pin 27 = RST

Pin 26 = VCC

Pin 25 = A7

Pin 24 = A6

Pin 23 = A5

Pin 22 = A4

Pin 21 = A3

Pin 20 = A2

Pin 19 = A1

Pin 18 = A0

Pin 17 = X

Pin 16 = X

Pin 15 = D13

Las conexiones realizadas serán de gran ayuda con respecto a las conexiones correspondientes al espadín de 11 que conecta con la regleta de sensores. Por lo tanto la imagen siguiente también servirá de referencia para hacer las conexiones de forma correcta, asignando a cada terminal de salida una salida correspondiente por lo que únicamente se debe dar click sobre la terminal y en "String" escribir el nombre a la terminal.

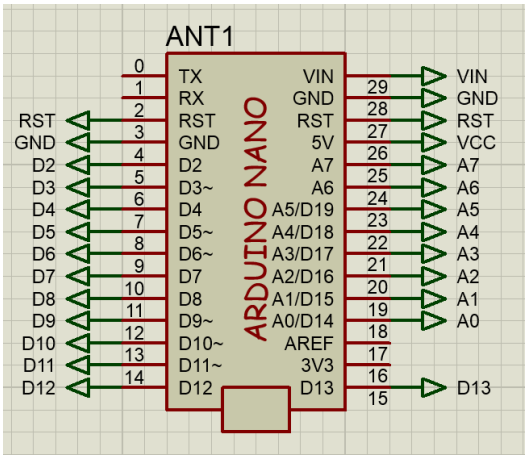


Imagen 98. Terminales para el Arduino Nano.

4.4.2 Conexión de terminales del arduino nano con el espadín de 11 (sensores)

Para las conexiones empleadas entre estos dispositivos será necesario partir de la barra de sensores TQR 8 de Pololu, por tanto se toma como referencia la conexión respeto a la distribución de pines de esta barra de sensores.

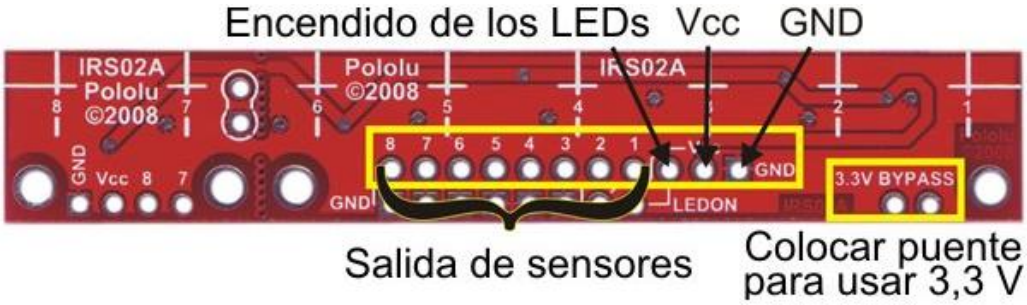


Imagen 99. Barra de sensores TQR 8 de Pololu.

Iniciando de izquierda a derecha los pines deberán ser los siguientes:

Pin 0 = A7

Pin 1 = A6

Pin 2 = A5

Pin 3 = A4

Pin 4 = A3

Pin 5 = A2

Pin 6 = A1

Pin 7 = A0

Pin 8 = D12

Pin 9 = VCC

Pin 10 = GND

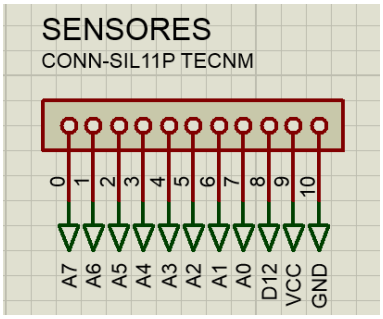


Imagen 100. Barra de sensores.

Una vez terminadas las conexiones se puede concluir que el robot está diseñado tanto para las regletas QTR 8 como las regletas multiplexadas de 13 y 16 sensores, para este caso la regleta a utilizar es una regleta de 16 sensores multiplexada de Ingeniero Maker y tanto esta regleta como la QTR 8 de Pololu tienen la misma distribución iniciando de derecha a izquierda con una tierra, un VCC, LED ON (D12), los pines de datos que son A0, A1, A2, A3 y el A4 como pin de lectura. Para el Arduino Nano el pin A4 funciona como pin digital D18 así como el pin D17, D16, D15 y D14 respectivamente. El pin A4 funcionará como único pin digital para las barras de sensores, ya sea de 12, 16 o QTR 8 y los pines A0, A1, A2 y A3 como pines digitales. El LED ON funcionará para activar el emisor de cada sensor y posterior a ello los dos sensores a la derecha restantes son de alimentación.

4.4.3 Conexión de terminales para el switch

Con las conexiones para los Switch ya realizadas lo siguiente es definir los terminales de salida, el voltaje de entrada (VIN) del primer Switch sale por el pin 3 hacia el Arduino Nano, el segundo Switch mandará voltaje al espadín de la turbina, que está representado por VIN T (Voltaje de entrada de la turbina) y el segundo pin de este espadín ira a tierra, de igual forma el pin 3 del 2do Switch va a VT (Voltaje de la turbina).

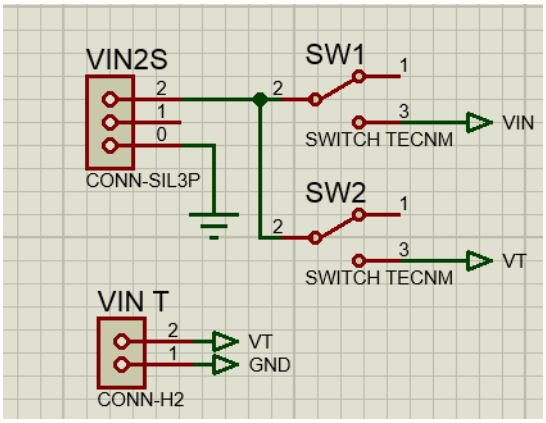


Imagen 101. Conexión de los Switch.

4.4.4 Conexión de terminales para el puente H TB6612FNG

Las conexiones del TB6612FNG son muy importantes puesto que ahí se definen los pines encargados de la regulación de velocidad para el motor y la regulación de la velocidad para la turbina. Para los motores serán designadas las velocidades positivas o negativas, o sea de avance o retroceso, lo cual le permitirá al seguidor ir hacia adelante y hacer los giros cambiando la velocidades de los motores en cuanto la barra de sensores indique que este se ha salido de la línea de la pista, por ello importante saber sobre las frecuencias por modulación de ancho de banda (PWM) de cada motor. Las conexiones para el TB6612FNG son las siguientes:

Pin 0 = VIN

Pin 1 = VCC

Pin 2 = GND

Pin 3 = MI2

Pin 4 =MI1

Pin 5 = MD1

Pin 6 = MD2

Pin 7 = GND

Pin 8 = GND

Pin 9 = D11

Pin 10 = D7

Pin 11 = D6

Pin 12 = VCC

Pin 13 = D5

Pin 14 = D4

Pin 15 = D3

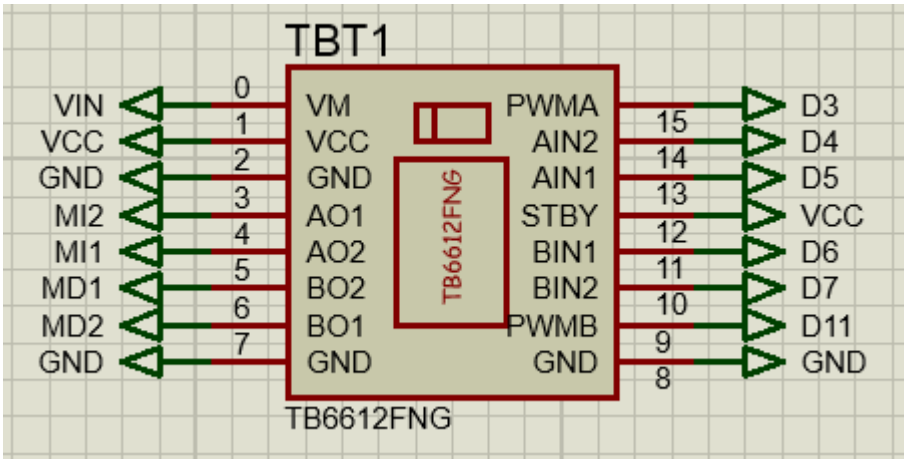


Imagen 102. Conexión de TB6612FNG.

La respuesta de frecuencia PWM varía según 2 factores:

- Motor

- Driver

Es necesario configurar ambos motores a una misma frecuencia, para tener la mejor respuesta posible de los motores y que el desempeño del seguidor de línea sea óptimo. La respuesta de frecuencia está basada en que: “Existe una frecuencia determinada para cada pin PWM, que es llama cuando se usa el comando analogWrite en ese pin. Y esta frecuencia determinada se puede cambiar a un valor tan alto como 65Khz y tan bajo como 30Hz usando solo un código de línea”.

4.4.4.1 Frecuencia para cada pin

Frecuencia PWM para D3 y D11: 490.20 Hz (El DEFAULT)

Frecuencia PWM para D5 y D6: 976.56 Hz (El DEFAULT)

Frecuencia PWM para D9 y D10: 490.20 Hz (El DEFAULT)

4.4.4.2 Los timer en pwm por hardware

Las funciones PWM por hardware emplean los Timer para generar la onda de salida. Cada Timer da servicio a 2 o 3 salidas PWM. Para ello dispone de un registro de comparación por cada salida. Cuando el tiempo alcanza el valor del registro de comparación, la salida invierte su valor.

Cada salida conectada a un mismo temporizador comparte la misma frecuencia, aunque pueden tener distintos Duty cycles, dependiendo del valor de su registro de comparación.

4.4.4.3 Asociación de timers y pwm

En el caso de Arduino Uno, Mini y Nano:

El Timer0 controla las salidas PWM 5 y 6.

El Timer1 controla las salidas PWM 9 y 10.

El Timer2 controla las salidas PWM 3 y 11.

Ahora bien ¿cuál sería la opción más adecuada para trabajar con los motores tipo Pololu?, de principio los pines a utilizar son: pin D3 y D11, debido a que ambos trabajan a una misma frecuencia determinada, pero no solo eso, sino que también en cuestión de Timer, el Timer2 que controla las salidas 3 y 11 están completamente libres, como es esto, bien pues el Timer0 además de controlar las salidas 5 y 6 también está ligado al reloj interno del Arduino Nano que trabaja con Delays y el Timer1 no solo controla la modulación por ancho de pulso (PWM) de los pines 9 y 10 sino que también está encargado de controlar la librería de los servomotores y esto ocasiona que al utilizar esas salidas para el ESC el sistema de prioridad a dichas librerías, sin embargo esto se puede corregir utilizando una librería especial para el ESC, para este caso sería ESC.h que regula la velocidad de la turbina, por lo tanto los pines 9 y 10 son los más convenientes para conectarlos a la turbina y poder trabajar con ella. Entonces esto da como resultado a las salidas 3 y 11 del Timer2 para regular la velocidad de los motores.

4.4.5 Conexiones de terminales para los motores

Los motores deberán ser identificados de tal forma en que se pueda distinguir entre el motor izquierdo y motor derecho, para ello con doble click sobre cada uno se puede cambiar el nombre con el que están registrados, después para el motor izquierdo se tiene que colocar iniciando del pin superior al inferior: Pin 2 = MI1 y Pin 3 = MI2 y para el motor derecho nuevamente iniciando del pin superior al inferior: Pin 2 = MD1 y Pin 3 = MD2.

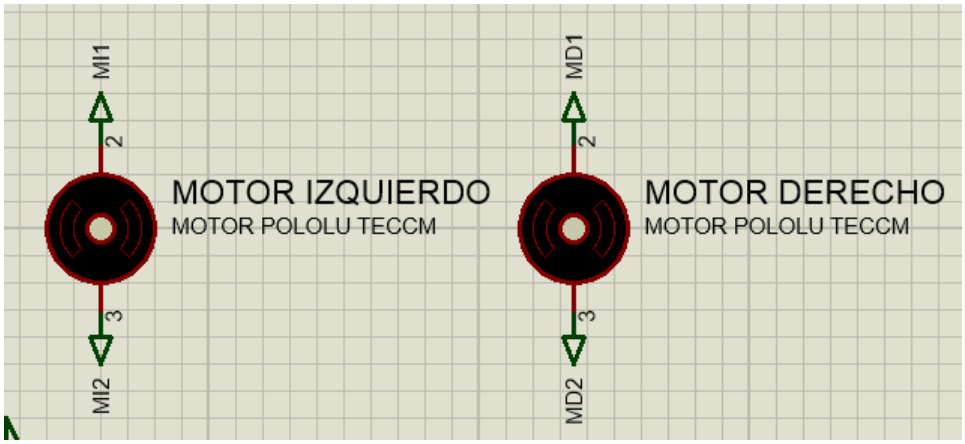


Imagen 103. Motores.

4.4.6 Conexión de terminales para el modulo arrancador

Para el modulo arrancador únicamente se debe colocar de manera descendente las siguientes salidas:

Pin 4 = X

Pin 3 = D2

Pin 2 = GND

Pin 1 = VCC

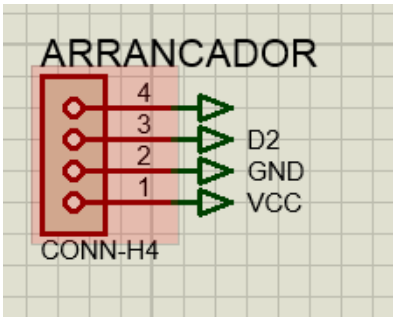


Imagen 104. Modulo Arrancador.

4.4.7 Conexión de terminales para el pulsador

La información del pulsador será enviada al pin D9

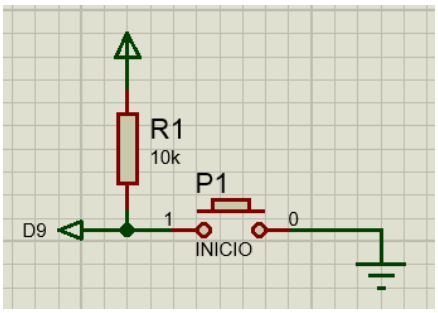


Imagen 105. Pulsador.

4.4.8 Conexión de terminales para el esc

Por último quedan las conexiones para el VCC para ello se utiliza un espadín de 3 y de forma descendente se colocan los terminales de salida de la siguiente manera:

Pin 2 = D10

Pin 1 = VCC

PIN 0 = GND

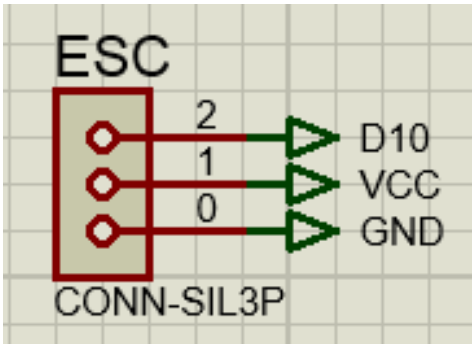


Imagen 106. ESC.

4.5 Diseño de placa (pcb)

El diseño de la placa será realizado en el PCB Layout de Proteus, utilizando el sistema inglés (Pulgadas) debido a que este ha sido el sistema con el que se trabajó y la mayoría de mediciones están dadas en pulgadas, por tanto se debe cambiar de sistema métrico a sistema inglés en la parte superior del espacio de trabajo, justo en la "m" encerrada en un círculo de color rojo. También se debe reducir el tamaño de la cuadricula del espacio de trabajo, para eso en la parte superior izquierda en el apartado "View" se debe seleccionar la opción "Snap 50 th" que es una relación de trabajo de 50,000 pulgadas.

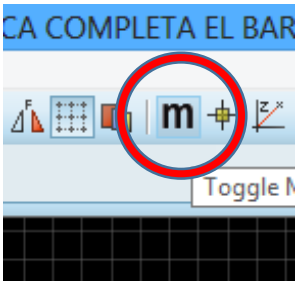


Imagen 107. Sistema Métrico a inglés.

Se deben colocar todos los empaquetados realizados para darles orden en cuanto a la distribución en la placa, iniciando por la turbina puesto que la turbina realiza un trabajo de succión, por tanto, para que esta succión se realice de manera uniforme lo ideal sería colocar la turbina en la parte central de la placa (PCB). Para ello se debe buscar la turbina en "Components Mode" con el nombre con el que fue guardado el empaquetado, y colocarlo justo en el centro del espacio de trabajo encima de la cruz azul que marca Proteus como su centro, cuidando que este lo más centrado posible y tenga las mismas distancias para todos sus lados. Se puede tener una referencia de la turbina centrada teniendo 6 cuadros por cada lado, tanto en superior, inferior, izquierda y derecha y todo con una View de Snap 50 th.

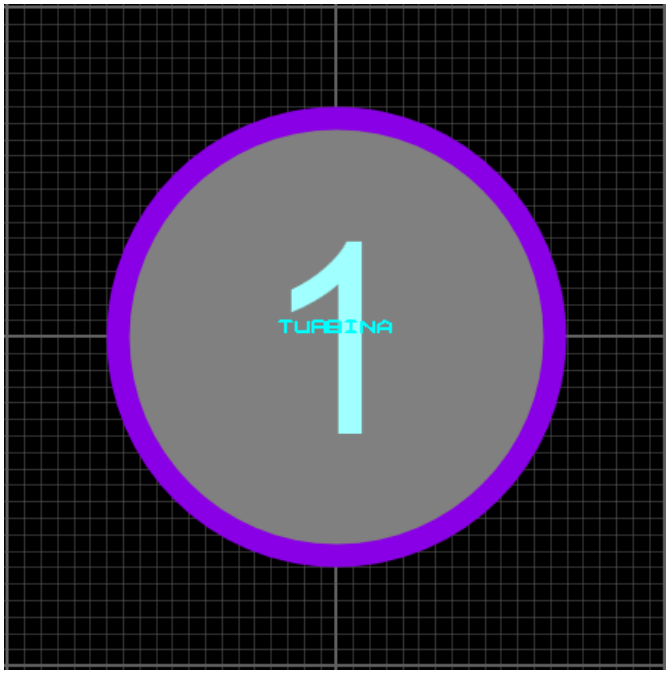


Imagen 108. Turbina centrada.

Una vez colocada la turbina de forma central se debe realizar una modificación en la medida del grosor del Pad de perforación (Circulo morado) puesto que toda el área gris del Pad representa la parte perforada, el área morada quedara metalizada una vez se realice la placa física. Para realizar este cambio se debe seleccionar nuevamente el Pad C500-60 y dar doble click sobre esta opción cambiando únicamente el de diámetro externo, dejándolo en 1.4 pulgadas.

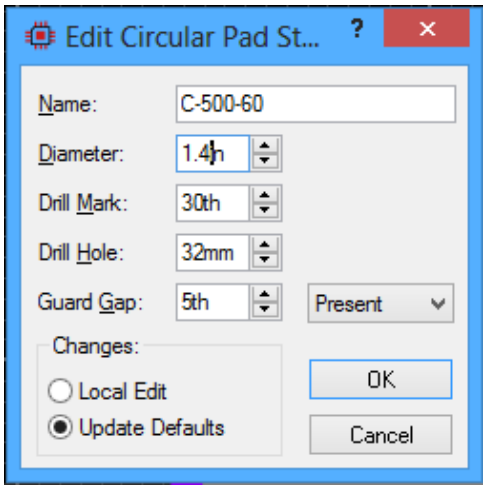


Imagen 109. Diámetro externo de la turbina.

El objetivo de esta modificación en el diámetro externo del Pad es que gane rigidez en esa área, para que al momento de funcionar la turbina, por la potencia que genere no permita un dobles en la placa.

Después de haber colocado la turbina en una posición central y revisar el grosor de su diámetro externo, se coloca el Arduino Nano a la izquierda de la turbina respetando la siguiente distancia entre ambos: 250 milipulgadas en horizontal, que con ayuda de la herramienta de medición marcada con flechas, se puede observar que del inicio de la turbina, la medición terminara a la mitad del Pad de perforación del pin 18 del Arduino Nano y tomando como referencia la línea en horizontal en color gris con un grosor más grande que el de las demás líneas del espacio de trabajo se debe tener una distancia de 300 milipulgadas casi llegando al conector USB que se dibujó como referencia para el Arduino Nano, dejando únicamente un espacio de un cuadro de 50 milipulgadas.

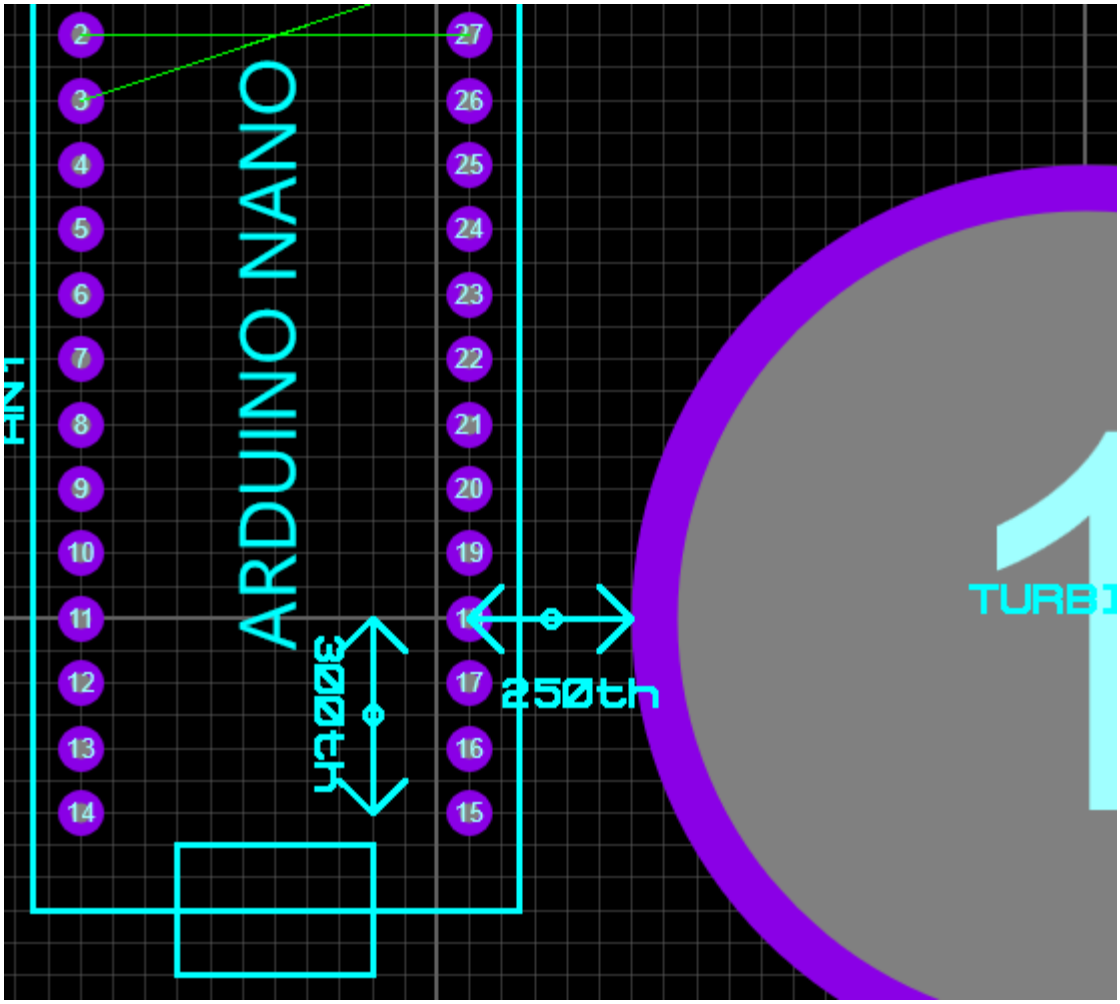


Imagen 110. Ubicación del Arduino Nano con respecto a la Turbina.

Todos los empaquetados deben salir del “Component Mode” puesto que esos componentes tienen las conexiones entre dispositivos realizadas anteriormente, por este motivo se pueden observar líneas en color verde del Arduino Nano siendo estas las conexiones mencionadas. Nuevamente se saca al TB6612FNG del “Components Mode” y se coloca a la derecha de la turbina, a una distancia de 350 milipulgadas en horizontal, dando justamente al Pad 12 del TB6612FNG y en vertical hacia abajo a una distancia de 275 milipulgadas dando justamente al inicio del Pad 15 del TB6612FNG. Para poder trabajar a 275 milipulgadas se debe ajustar el Snap a 25 th.

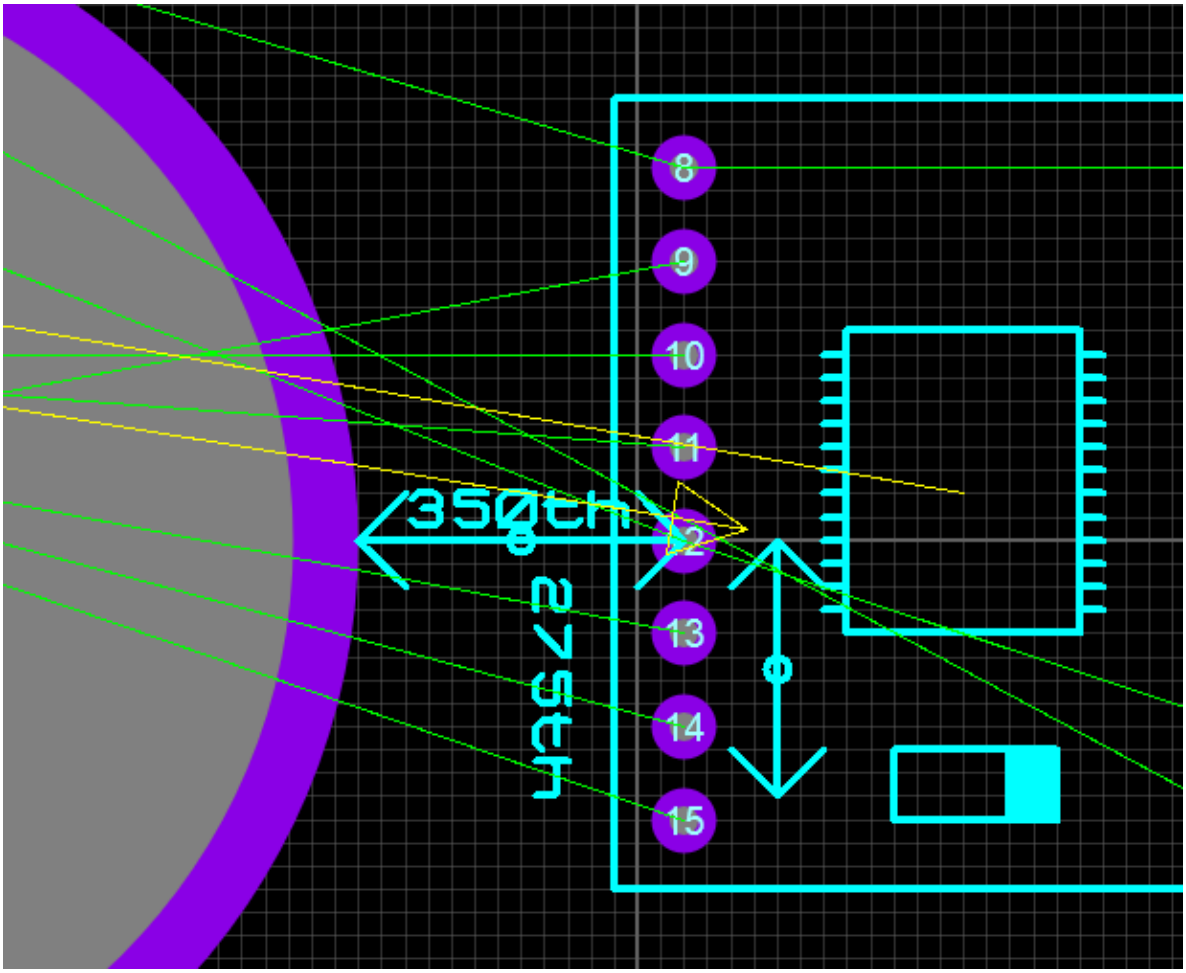


Imagen 111. Ubicación del TB6612FNG con respecto a la Turbina.

Después de colocar el TB6612FNG lo siguiente es colocar el conector para los sensores, siempre recomendando que este conector en la medida de lo posible tenga una conexión directa, debido a que el dato de los sensores es de lo más importante en cuanto al seguidor de línea puesto que con ello la barra de sensores multiplexada enviara los datos necesarios para que todo el sistema funcione de forma adecuada. Por tanto se selecciona la opción "sensores" dentro del "Component Mode" para colocar el conector para la barra de sensores, este conector debe ser ubicado en la parte superior de la placa, justo por encima de la turbina, tomando como referencia al pin 5 para centrarlo, entonces tomando como referencia la línea vertical gris un poco más gruesa que las demás, se

central el conector de sensores en el Pad 5 con una distancia de 550 milipulgadas partiendo de la turbina al centro el Pad.

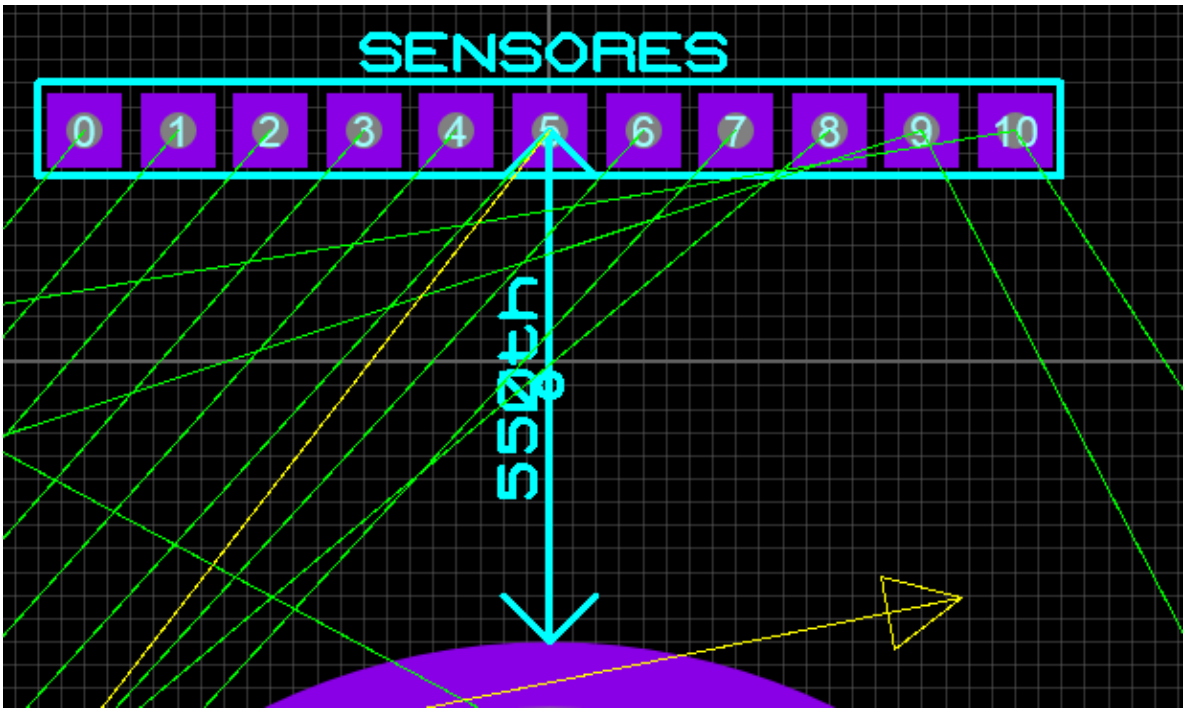


Imagen 112. Ubicación del conector de sensores con respecto a la Turbina.

Con el conector de los sensores colocado, se debe realizar la conexión de los datos que necesita el conector para los sensores, como se puede ver, las líneas verdes indican las conexiones que existen entre el conector de sensores con los pines analógicos del Arduino Nano siendo la primera conexión realizada por la importancia de datos.

Utilizando la herramienta “Track Mode” se selecciona la opción “Default” para verificar que la opción “Width” se encuentre en 20 th o sea 20 milipulgadas, que en caso de tener otro valor lo único por hacer es realizar el cambio a 20 th, con la finalidad de tener un ancho de vía de 20 milipulgadas, lo siguiente es dar click en “OK” y comenzar a unir el pin 18 del Arduino Nano con el pin 7 del conector de sensores, esto es muy fácil porque prácticamente el programa muestra el

camino de unión que se debe seguir, teniendo trazado un camino de inicio a fin. Se debe prestar atención a las vías para evitar que choquen entre sí o que choquen con algún otro componente, este proceso se debe realizar con los demás pines del Arduino Nano en forma ascendente hasta el pin 25 y el pin 0 del conector de sensores. Las vías se marcan en color azul debido a que las conexiones realizadas son en la parte inferior de la placa, estás serán las vías de cobre que formaran parte de la placa en estado físico, en la parte inferior de la pantalla se encuentran las palabras Bottom Copper precedidos de un rectángulo de color azul marino que indican las conexiones de vía inferior, si por otro lado se pretenden realizar las conexiones en la parte superior se debe seleccionar la opción Top Copper que cambiara las vías a un color rojo y las vías a la parte superior de la placa.

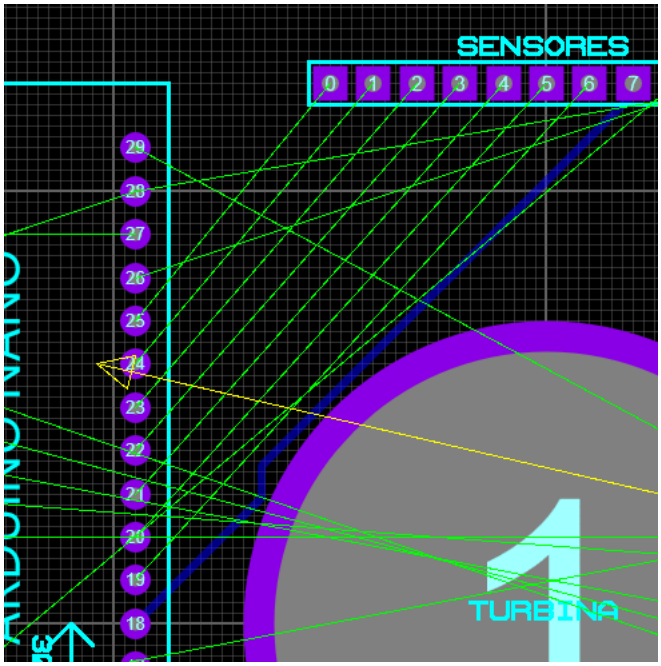


Imagen 113. Conexiones de vía entre el Arduino y el conector de sensores.

Todas las conexiones realizadas entre el Arduino Nano y el conector de sensores deben ser acomodados a modo de ahorrar la mayor cantidad de espacio posible, porque para los componentes restantes también se aplicaran estas conexiones.

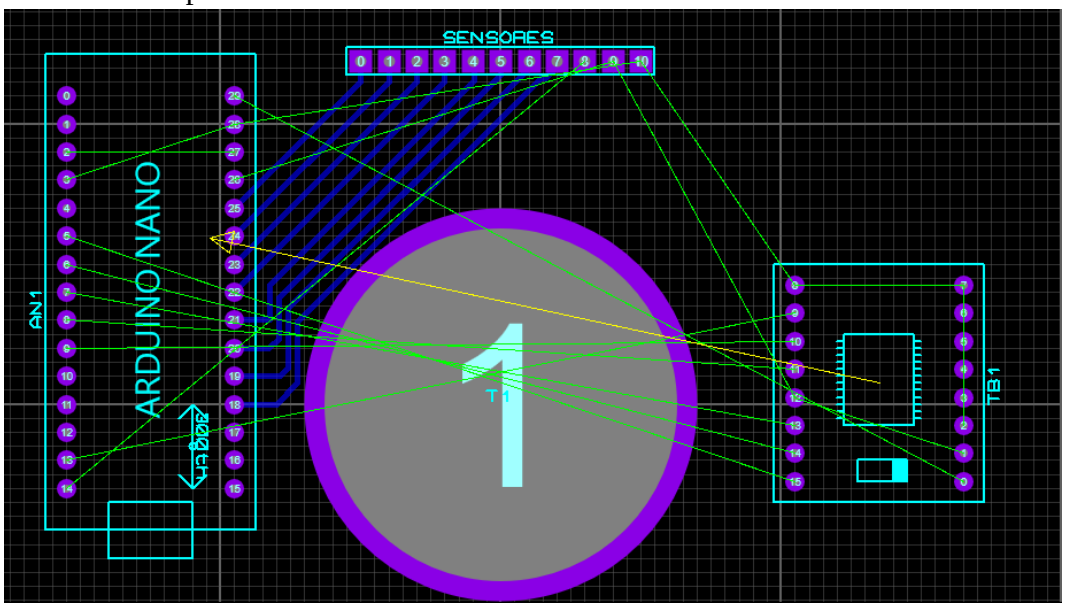


Imagen 114. Conexiones de vía entre el Arduino y el conector de sensores completas.

En el modelado 3D (3D Visualizer) se pueden observar las vías realizadas entre dispositivos aún sin tener una placa definida, esta vista puede ser tanto inferior como superior.

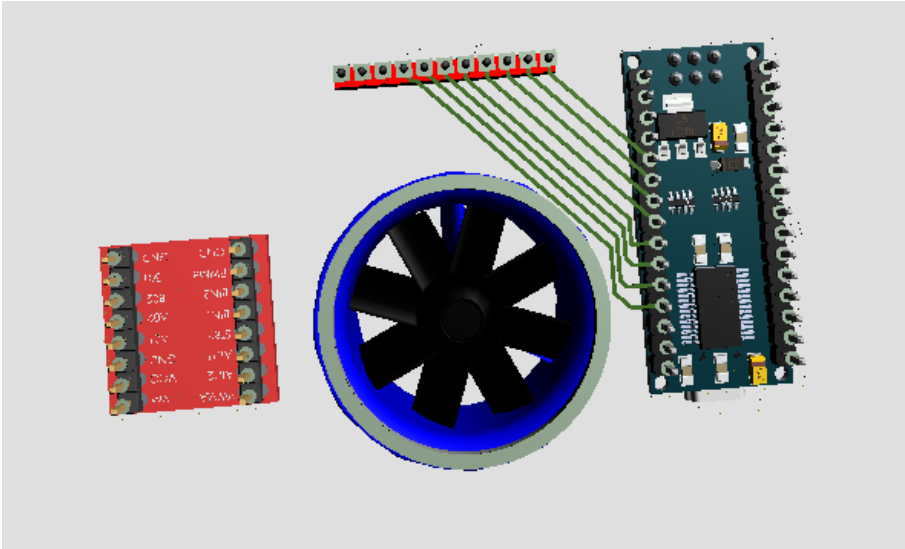


Imagen 115. Conexiones de vía entre el Arduino y el conector de sensores en 3D (vista inferior).

Otra conexión importante a realizar es la conexión para regular la velocidad de los motores, pero para realizar esta conexión con entre estos dos pines primero se deben colocar los motores en los extremos traseros de la placa, de tal forma en que queden a los laterales. Para esto se debe sacar un primer motor del "Component Mode" y colocarlo en la parte inferior izquierda a 2450 milipulgadas del centro, con ayuda de una flecha de medición se determinara la distancia. Justo al final de esta flecha debe terminar el motor, y para la parte vertical, el motor debe estar ubicado a 450 milipulgadas de arriba hacia abajo de la barra central horizontal. Teniendo como referencia que el Pad número dos del motor queda centrado a una esquina del conector USB del Arduino Nano. Este mismo proceso se realiza para el motor derecho, sacando otro motor del "Component Mode" para después colocarlo, como queda con el mismo sentido al motor izquierdo se debe rotar dos veces en sentido anti-horario para que quede invertido al primer motor. De igual forma, la distancia del centro al final del motor deberá ser de 2450 milipulgadas y en vertical debe tener una distancia de 450 milipulgadas, la misma que el motor izquierdo quedando a la misma altura. Posterior a la colocación de los motores se pueden comenzar a realizar los bordes de la placa que va a contener todos los componentes, además, con ayuda del visualizador 3D se puede tener una perspectiva de la posición de los motores.

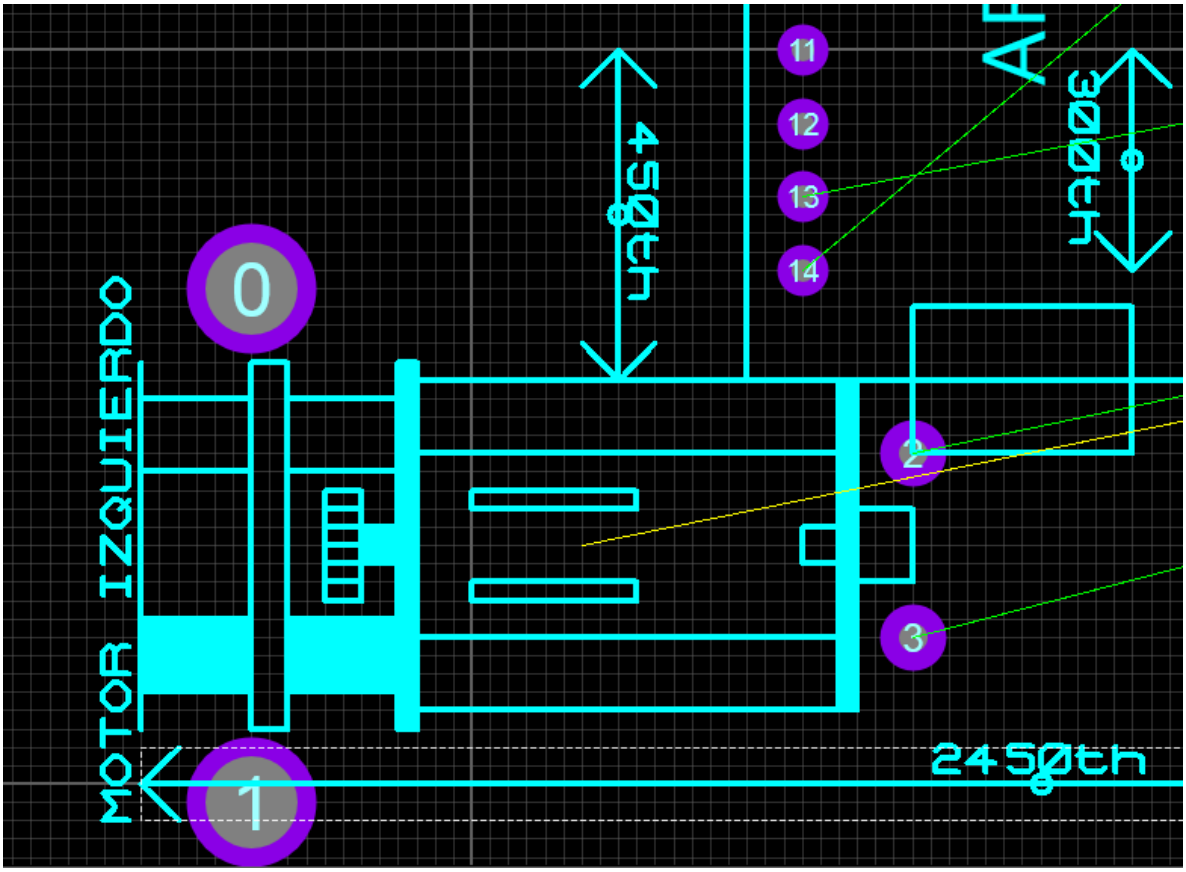


Imagen 116. Ubicación del motor izquierdo.

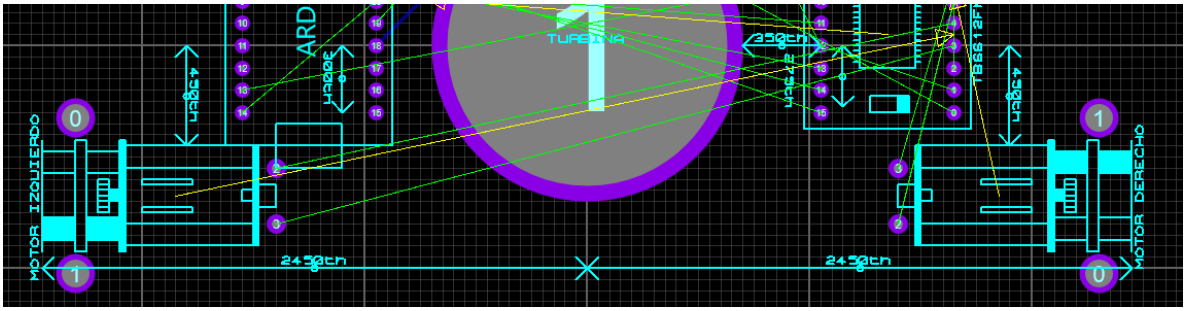


Imagen 117. Ubicación de ambos motores.

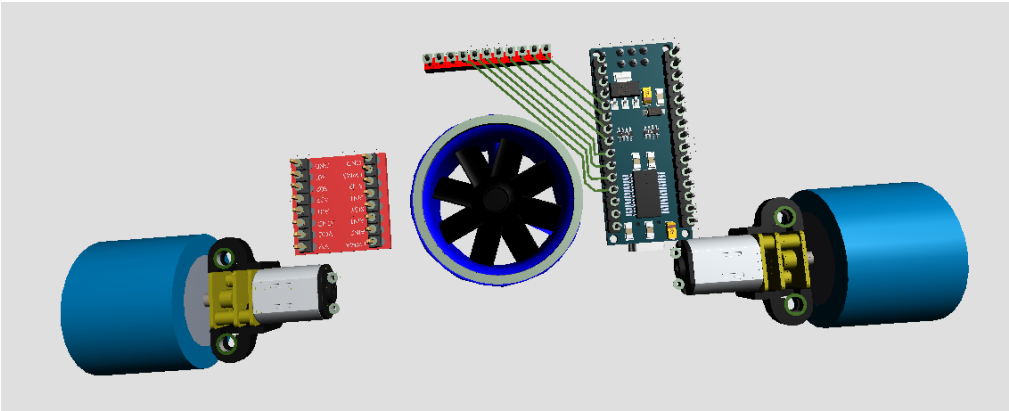


Imagen 118. Ubicación de ambos motores 3D.

4.5.1 Bordes de la placa

Para realizar los bordes de la placa se debe seleccionar la herramienta de líneas, esta herramienta se encuentra en la parte lateral izquierda de la pantalla de Proteus y responde al nombre de “2D Graphics Line Mode”, después de seleccionarla en la parte inferior se debe cambiar la opción de “Top Silk” a “Board Edge” que se muestra en color amarillo. La siguiente es una lista de pasos para realizar los bordes de la placa.

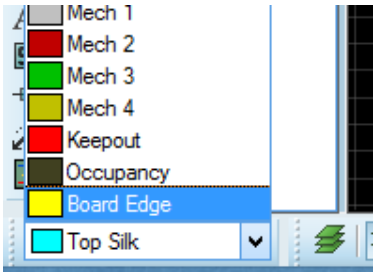


Imagen 119. Board Edge.

1. Lo primero a realizar para el desarrollo de los bordes será una línea recta de 1300 milipulgadas, ubicada sobre la parte trasera del motor izquierdo, iniciando a 225 milipulgadas de donde empieza el motor viéndolo de abajo hacia arriba, justamente la línea vertical de 1300 milipulgadas debe quedar entre el pin número 10 y 9 del Arduino Nano.

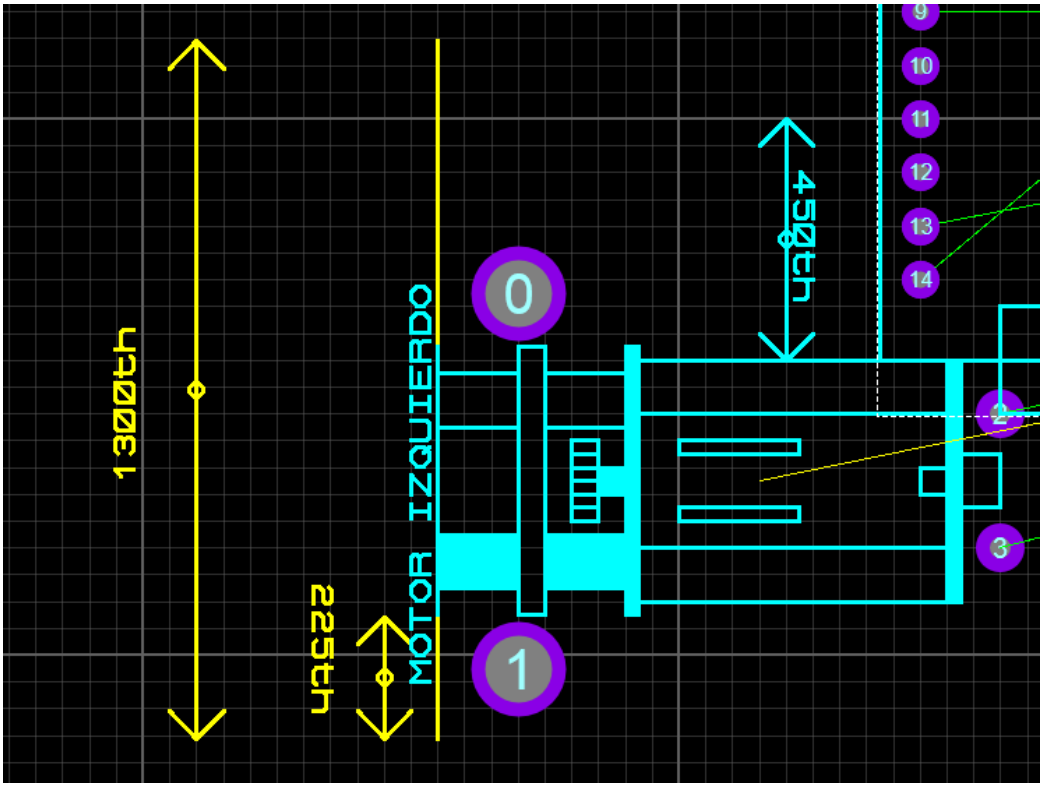


Imagen 120. Línea vertical.

2. Se realiza una segunda línea en diagonal partiendo de la línea de 1300 milipulgadas por la parte inferior, esta línea en una medición recta en vertical debe estar colocada a 125 milipulgadas de donde termina la primera línea, finalizando prácticamente a la mitad del Pad de perforación inferior del motor izquierdo.

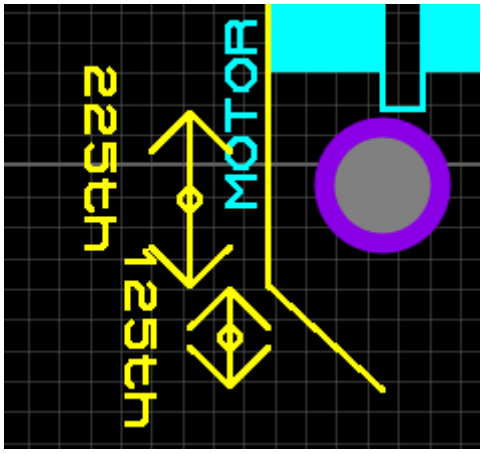


Imagen 121. Primer línea diagonal.

3. Partiendo del final de la línea en diagonal se debe colocar una línea en horizontal con una distancia de 850 milipulgadas.

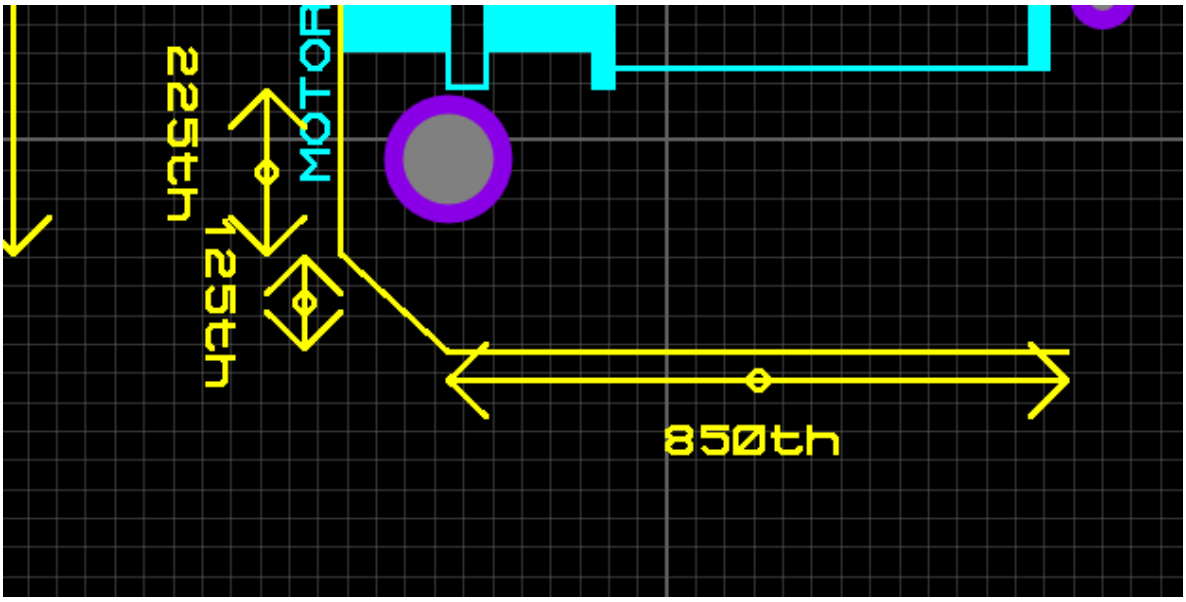


Imagen 122. Primer línea horizontal.

4. Nuevamente se realiza una segunda línea diagonal que debe tener 100 milipulgadas de base y 100 milipulgadas de altura, para concluir la parte inferior de la placa con una línea horizontal de aproximadamente 1380 milipulgadas, o justo a la mitad de la turbina que sería la mitad de la placa general, ¿por qué 1380 milipulgadas es un aproximado? porque dependiendo de qué tan precisos sean al realizar todas las medidas, se puede tener un margen de error, este margen de error puede ser de 10 milipulgadas hasta 20 milipulgadas, y no afectaría en gran medida el desarrollo de la placa, siempre y cuando esta última línea horizontal termine justo a la mitad, en dirección a la turbina. Estas dos últimas líneas concluyen la parte inferior.

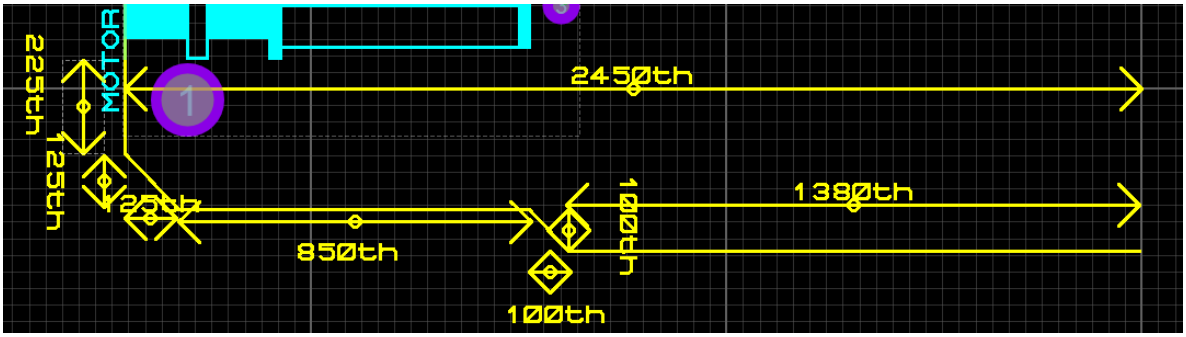


Imagen 123. Segunda línea diagonal y horizontal.

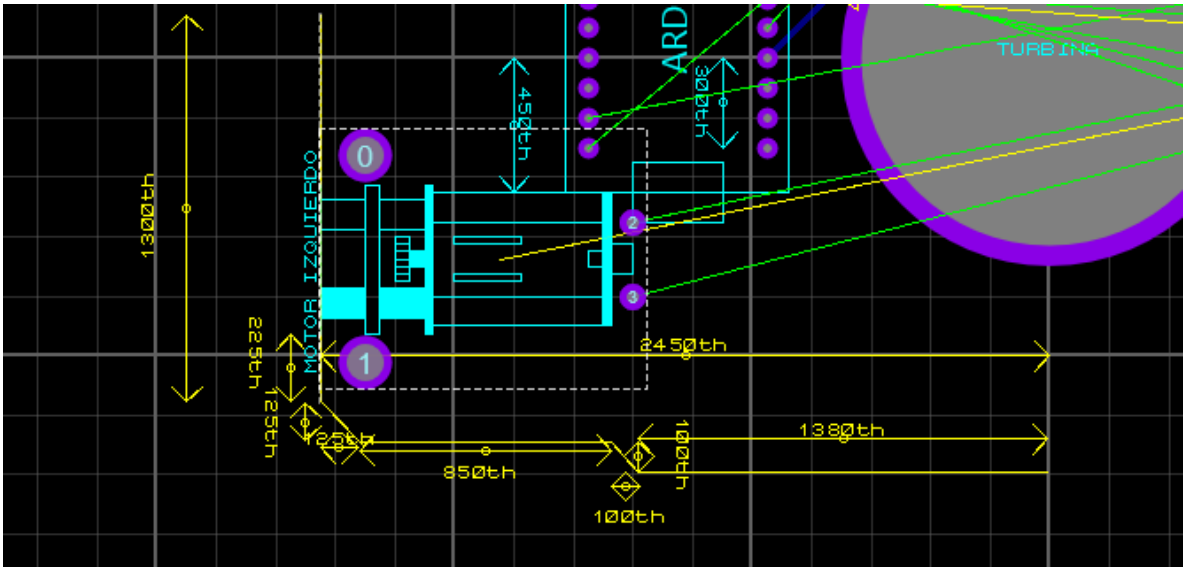


Imagen 124. Parte inferior de la placa.

5. Ahora para la parte superior de la placa, partiendo de la parte superior de la línea recta inicial de 1300 milipulgadas se colocara una línea en diagonal con 300 milipulgadas de base y 375 milipulgadas de altura.

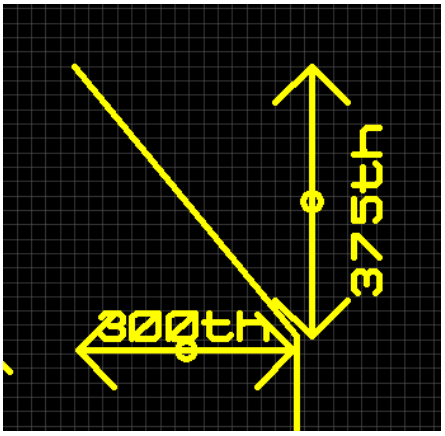


Imagen 125. Primer diagonal superior.

6. Nuevamente se coloca una línea vertical de 650 milipulgadas y posterior a ello se coloca otra línea en diagonal con base de 150 milipulgadas y una altura de 150 milipulgadas.

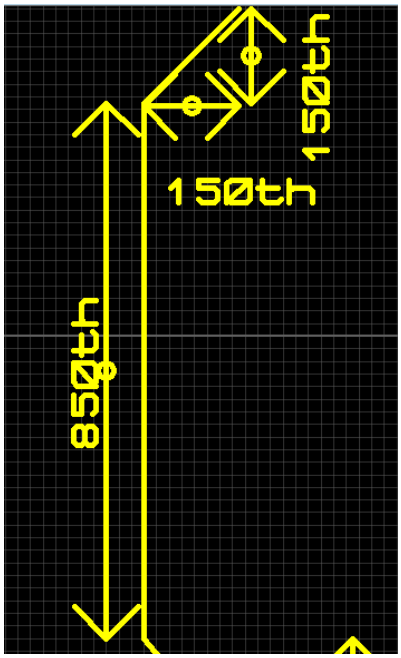


Imagen 126. Línea vertical y diagonal.

7. Se coloca una línea en horizontal de 1475 milipulgadas y después una línea diagonal de 100 milipulgadas de base y 175 milipulgadas de altura, para finalizar la parte superior con una línea que dé al centro de la placa en horizontal de aproximadamente 1030 milipulgadas.

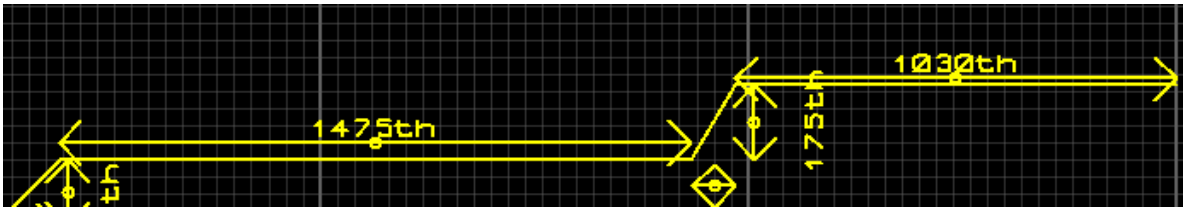


Imagen 127. Parte superior de la placa.

Lo único que resta por realizar es borrar todas las líneas de medición para después con la tecla "ctrl" seleccionar cada una de las líneas de borde de la placa y con click derecho seleccionar la opción Copy To Clipboard para copiar los bordes y pegarlos (Paste To Clipboard) en la parte derecha. Es muy importante que al colocar la nueva parte de los bordes se tenga cuidado de colocarlos en un espacio totalmente libre de cualquier componente para evitar que se encimen, después de colocarlos se seleccionan todos los bordes de la derecha y con click derecho se selecciona la función espejo (X- mirror) para por último unir la parte izquierda con la derecha.

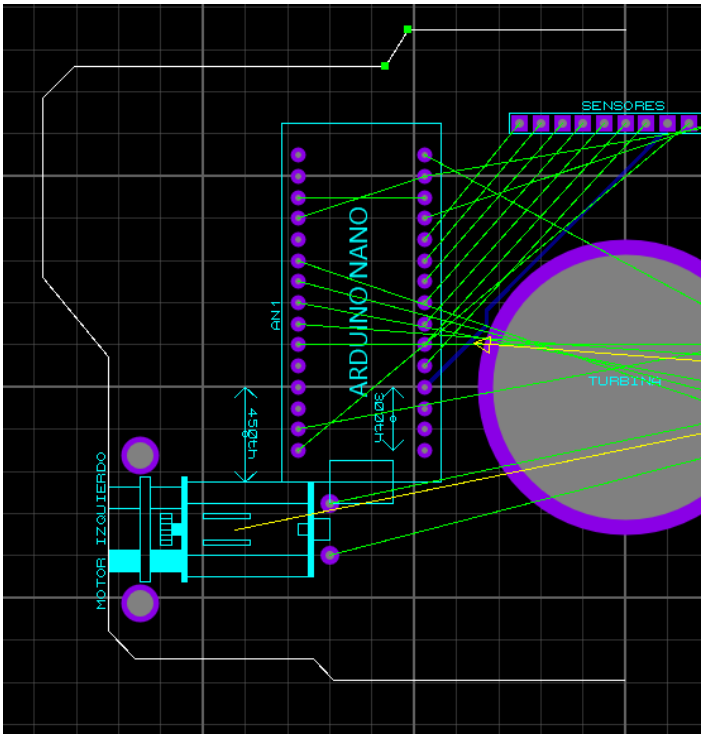


Imagen 128. Selección de la parte izquierda.

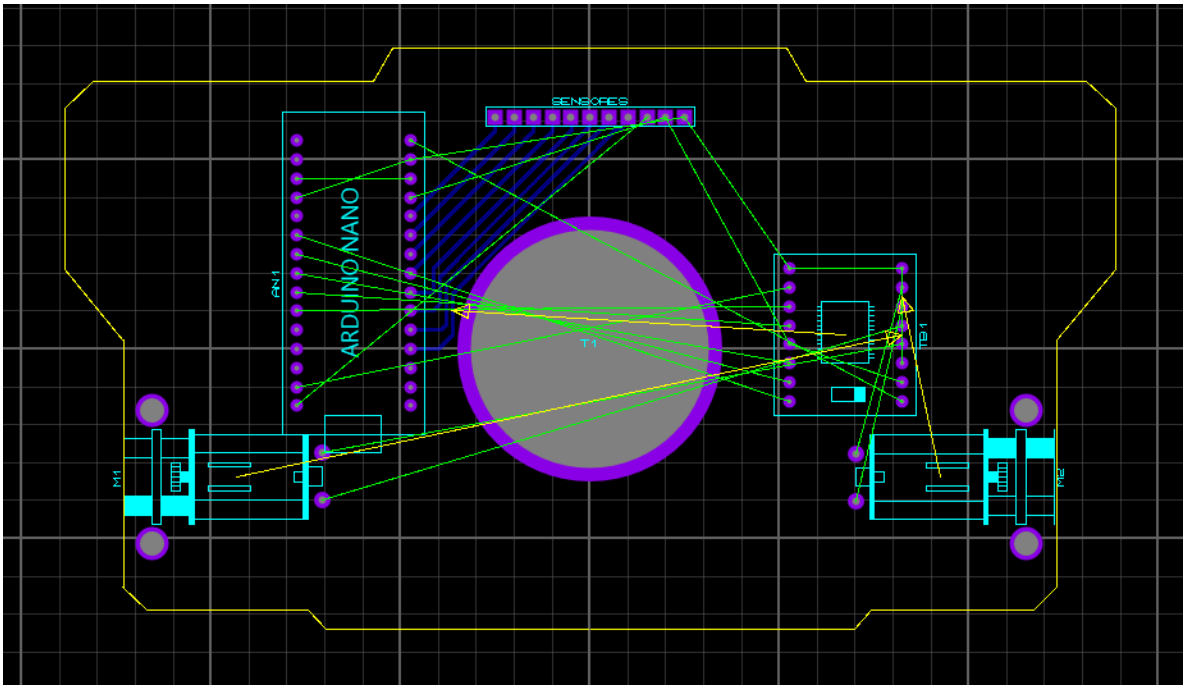


Imagen 129. Bordes completos.

Ahora con los bordes de lado izquierdo y derecho unidos, han quedado delimitados los bordes de la placa y esto se puede ver con ayuda del Visualizador 3D.

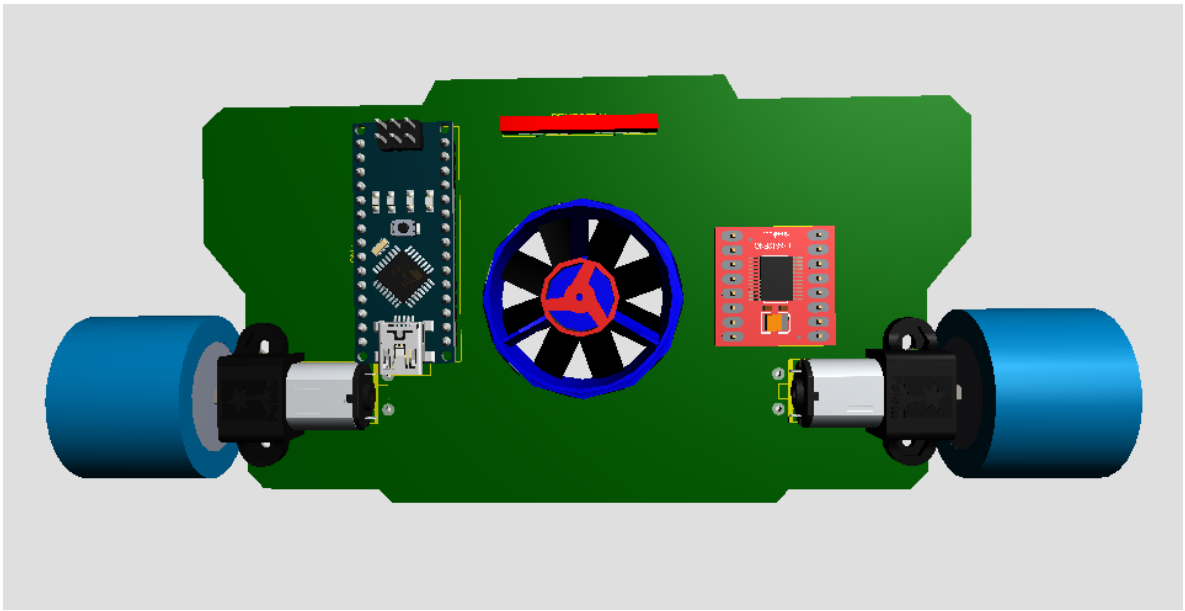


Imagen 130. Placa parte superior

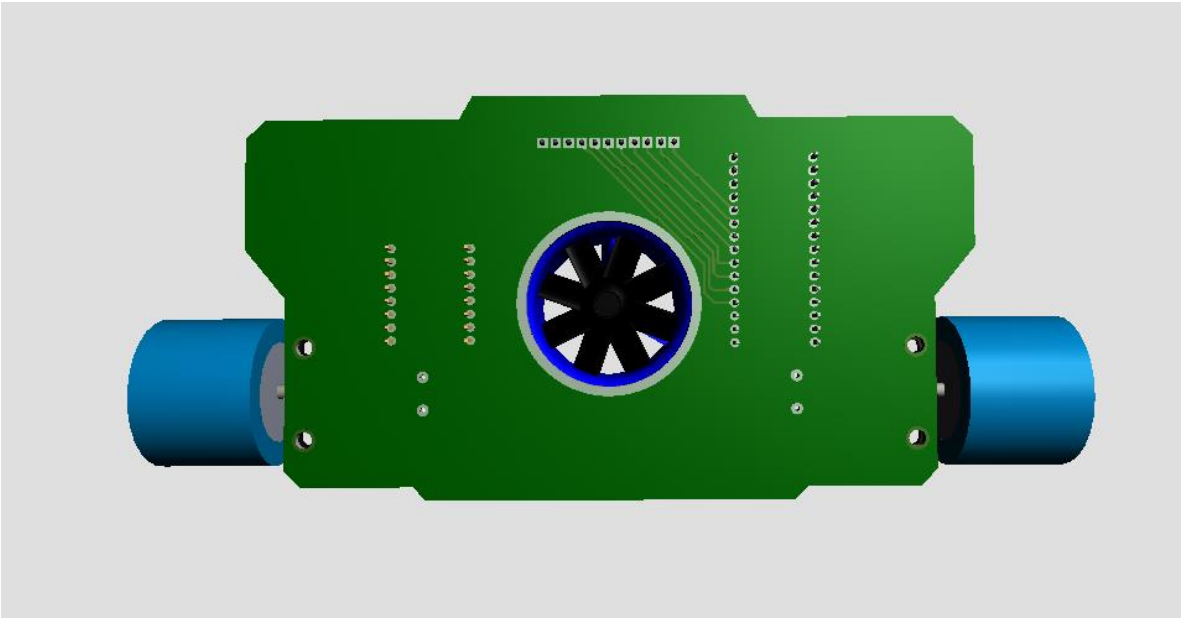


Imagen 131. Placa parte inferior.

Ahora que está completamente delimitado el espacio de trabajo de la placa, se pueden terminar de insertar los demás dispositivos que componen al seguidor de línea, iniciando por el arrancador, colocándolo en la parte superior izquierda de la placa, justo por encima del Arduino Nano. Una vez colocado, con click derecho debe ser rotado dos veces para que el pin 1 este de lado derecho y el pin 4 de lado izquierdo, con el fin de realizar las conexiones de manera más sencilla.

Imagen 132. Arrancador.

Posteriormente se coloca una resistencia de 10 k para después rotarla una vez en sentido horario, nuevamente por comodidad al momento de realizar conexiones. Esto se puede comprobar con las líneas verdes que salen de los componentes, estando lo menos enredadas posible.

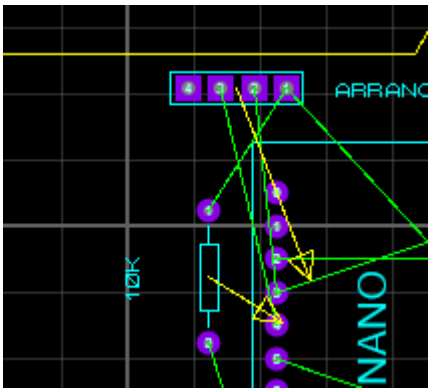


Imagen 133. Resistencia.

Se coloca tanto la fuente de alimentación para el ESC como los dos Switch justo por encima del puente H TB6612FNG en la parte derecha de la placa, para encender la turbina y el Arduino, cabe mencionar que estas posiciones para estos dispositivos no son definitivas, pues se podrán mover con el fin de la comodidad para colocar las vías que conectarán todos los dispositivos.

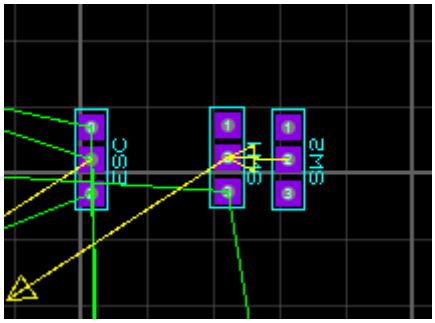


Imagen 134. ESC, alimentación y Switch.

En la parte inferior izquierda de la placa se coloca al pulsador, y en la parte inferior derecha se coloca el conector de alimentación para el sistema en general, que será de la batería.

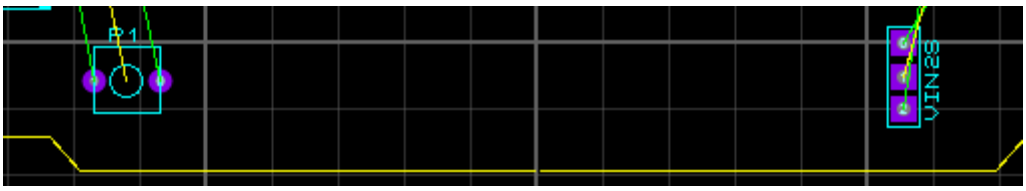


Imagen 135. Pulsador y fuente de alimentación.

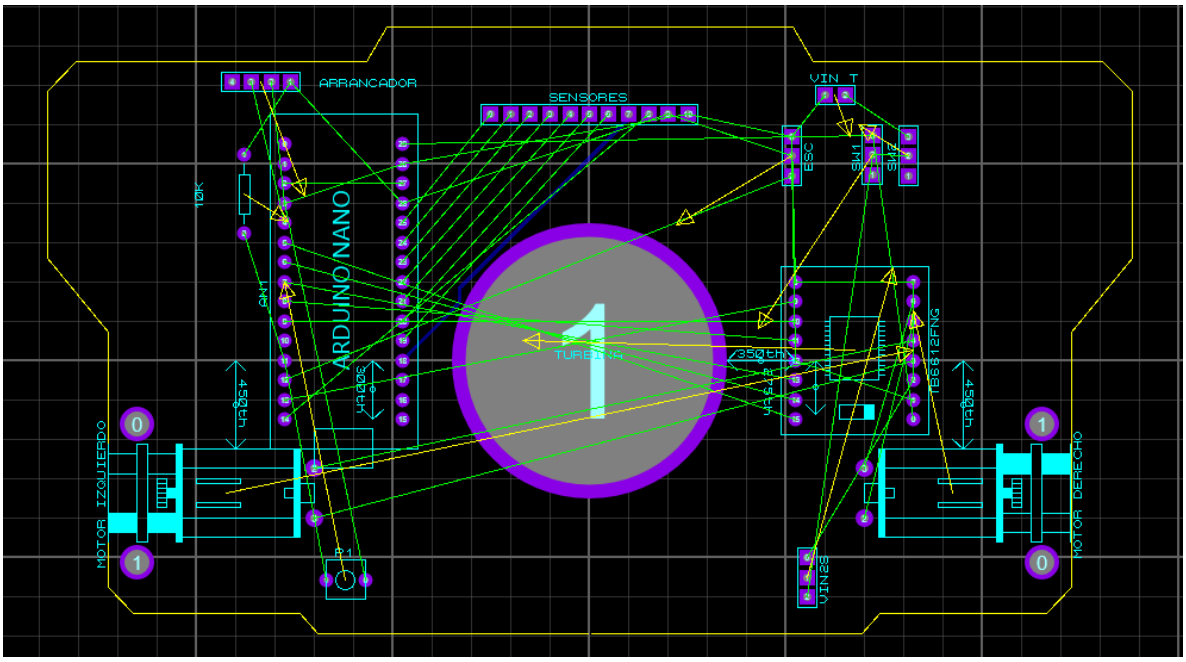


Imagen 136. Dispositivos en posición general para la PCB.

4.5.2 Huecos para la placa

Previo a realizar más conexiones entre vías es necesario realizar antes las perforaciones que permitan a los rieles unir y sostener a la barra de sensores junto con la PCB del seguidor, estas perforaciones serán en la parte superior e inferior de la placa. En la parte superior se realizarán 8 perforaciones, 4 por cada lado, teniendo como finalidad 2 cosas:

1- Poder acortar y o alargar la distancia entre la placa y la barra de sensores, ya sea para que la barra de sensores este en una posición que favorezca su comportamiento en pista, como la longitud

total del seguidor, debido a que en carreras profesionales este seguidor no debe pasar de 25 cm de largo x 20 cm de ancho, evitando incumplir con esta longitud máxima.

2- Existen diversos tipos de barras de sensores multiplexadas, desde 8 hasta 16 sensores, y por tanto también de diferentes marcas, generando que cada marca tenga características específicas, por ejemplo: La barra de sensores de Ingeniero Maker tiene ciertas características distintas a la barra de sensores de Phoniex. Esto puede deberse a diversos factores, que abarcan desde componentes, diseño de la empresa, resistencia, consistencia, desempeño, etc. Por tanto con las siguientes perforaciones la placa será compatible tanto con una barra de sensores de Ingeniero Maker como de Phoniex, y eso se verá a continuación:

Primero se debe seleccionar el tipo de Pads de perforaciones que se van a utilizar, en la barra de herramientas se debe seleccionar “Round Througth-hole Pad Mode” (herramienta para círculos) eligiendo la opción C180-M3, que es una medida correcta para los pernos o tornillos que realizaran la sujeción de la barra de sensores. Una vez elegido el tipo Pad de perforación establecido, lo siguiente por hacer es una línea de medición de 5120 milipulgadas justo por encima de la placa, centrada por el medio.

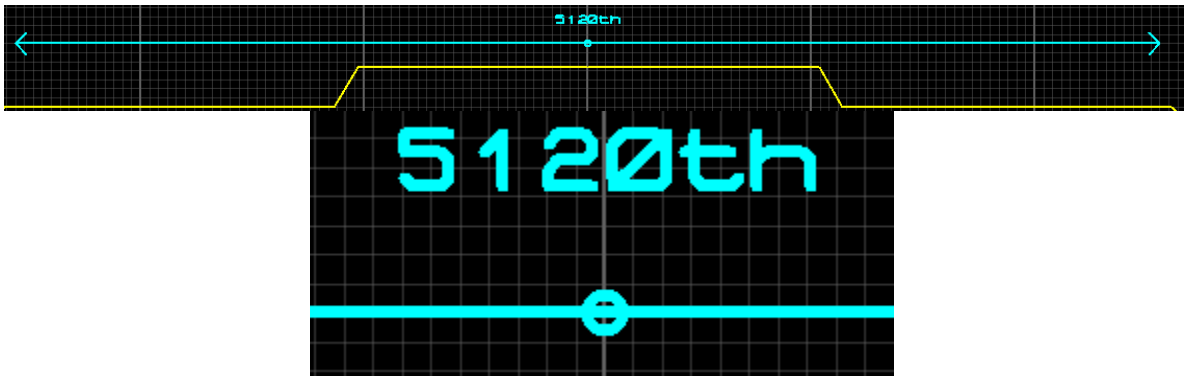


Imagen 137. Medida para la ubicación de las perforaciones para la barra de 16 sensores.

Estas medidas funcionan como un estándar para trabajar tanto con una barra de sensores como otra, que en el caso particular del proyecto la barra de sensores de Ingeniero Maker es la utilizada, pero también se puede utilizar otra medida para barras de sensores más pequeñas, que bien pueden ser de 12 o 13 sensores, y de igual forma las medidas que se utilizarían para estas barras de sensores serían las siguientes: 4250 milipulgadas.

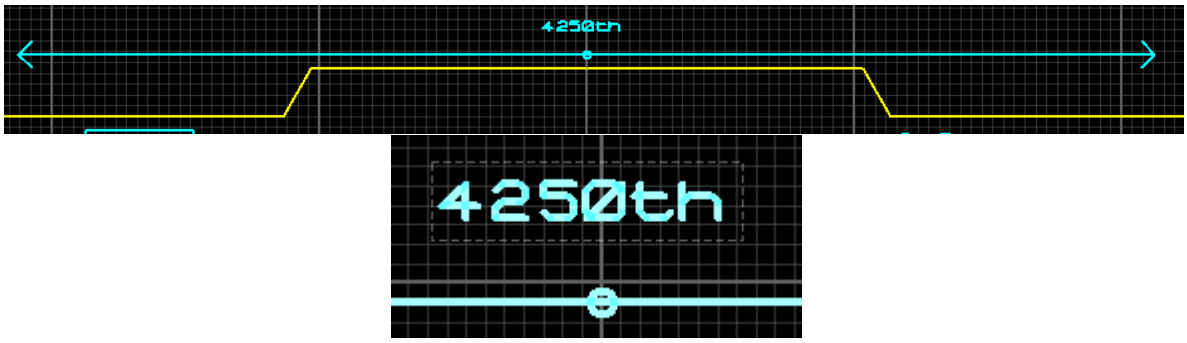


Imagen 138. Medida para la ubicación de las perforaciones para una barra menor a 16 de sensores.

Una vez aclarada la información entre los Pads de perforación es importante mencionar que para el proyecto se han trabajo con ambas distancias de perforación, sin embargo puede resultar innecesario, en un caso donde se tiene una barra de sensores establecida para trabajar, ya sea de 16 o menos sensores, resulta no ser de utilidad perforaciones de distancia menor para una barra que no se utilizara, por lo que se recomienda definir una barra de sensores para trabajar y en base a ello hacer las perforaciones con la distancia correspondiente. ¿En qué caso sería eficiente tener ambas medidas de Pads? en caso de utilizar diferentes tipos de barras de sensores, facilitando la modificación de la posición de las barras en la PCB.

Por ahora, el enfoque solo será explicar la colocación de los Pads con la primera medida, pero cabe recalcar que la PCB del proyecto cuenta con ambas medidas, por lo mencionado con

anterioridad, si se desea utilizar ambas medidas solo se debe seguir el mismo procedimiento para realizar las medidas menores o mayores dependiendo del caso.

En caso de utilizar una barra de sensores QTR8 de Pololu la distancia necesaria para colocar los Pads de perforación corresponden a 1810 milipulgadas, que en caso de requerirlas se pueden colocar. En el caso del proyecto si se realizaron pero no se utilizaron.

Una vez aclaradas las perforaciones de la PCB lo siguiente es realizarlas. Se debe colocar un Pad C180-M3 por cada extremo de la primera medida (5120 milipulgadas) de la siguiente manera: Se coloca cada Pad justo por encima de las flechas que aparecen con la herramienta de medición, cuidando que se centren lo mejor posible.

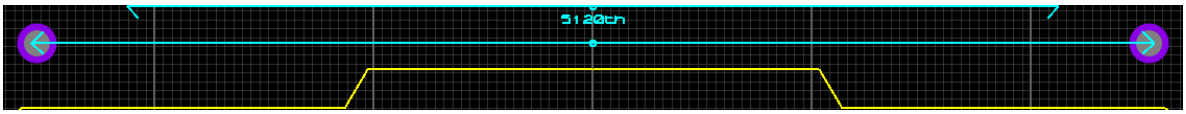


Imagen 139. Distancia para perforaciones de Pads.

Una vez colocado los Pads a los extremos se debe eliminar la línea de medición, después apoyados de la tecla “Ctrl” y el mouse se debe seleccionar un Pad seguido del otro, sin soltar la tecla “Ctrl” entre la selección para posteriormente con la herramienta de “Copy Block” de la parte superior, se copien y muevan ambos Pads para colocarlos sin afectar a los demás dispositivos en la parte de la placa.

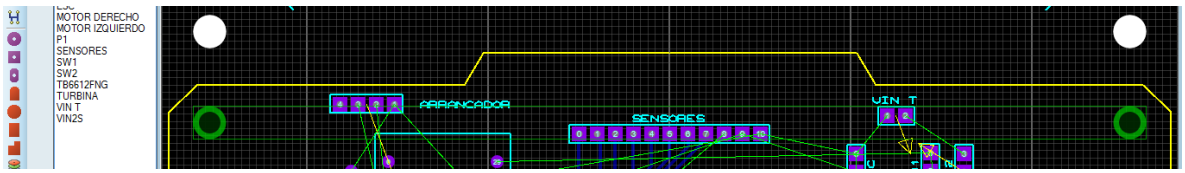


Imagen 140. Pads en placa.

Después de colocar los 2 primeros Pads sebe realizar el mismo procedimiento y debajo de los primeros colocar 3 más por lado, con la finalidad de que al momento de colocar los tornillos que sujetan las extensiones para la barra, se puedan colocar cómodamente uno o dos tornillos por lado. Estas perforaciones tendrán una distancia de 215 milipulgadas entre ellas.

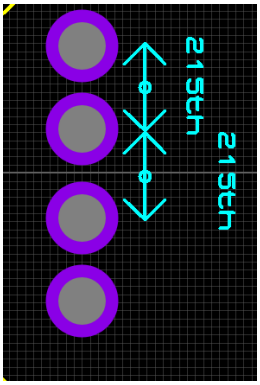


Imagen 141. Distancia entre los Pads de perforación de la placa

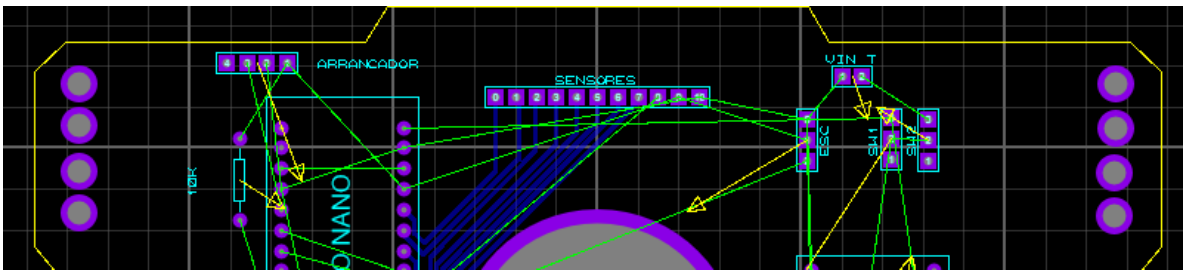


Imagen 142. Resultado de las perforaciones para la barra de sensores en la placa.

4.5.3 Conexión de vías entre dispositivos

La conexión de vías entre los dispositivos ya fue mencionada con anterioridad, y esto consta de sacar vías en la parte inferior de la placa para que de forma interna los dispositivos estén interconectados entre sí. En un inicio se realizó la conexión entre el lado derecho del Arduino Nano y la conexión para la barra de sensores, ahora las conexiones serán para los dispositivos restantes.

Las conexiones siguientes serán entre los motores y el puente H o TB6612FNG, entre los pines de los PWM y los motores, etc., es importante resaltar que estás conexiones se deben colocar siguiendo el procedimiento y también el criterio personal, cuidando siempre que las vías de conexión quepan en el espacio de la placa.

Se deben seleccionar las vías de trabajo por defecto en la herramienta “Track Mode” con un grosor de 20 milipulgadas (también un valor por defecto) después se debe conectar el pin 9 del puente H que corresponde al PWMB hacia el pin D11 del Arduino Nano. El pin número 15 del puente H que corresponde al PWMA hacia el pin D3 del Arduino Nano. Al momento de seleccionar un pin el programa trazara una línea a seguir para conectar con el pin correspondiente, en caso de no comprender las conexiones se recomienda revisar el diagrama de conexiones de salida entre dispositivos de la página 93 o el Schematic Capture. Un punto importante de las conexiones es no generar ángulos de 90° cuidando que no existan esquinas.

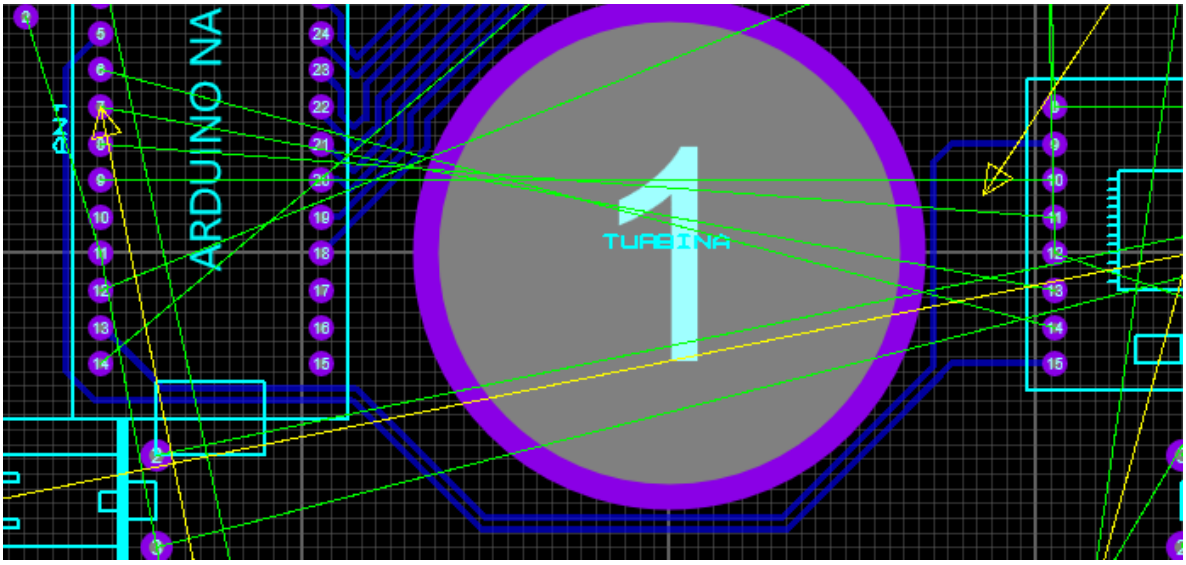


Imagen 143. Conexiones PWM.

La siguiente conexión de datos es entre el Arduino Nano y el módulo arrancador, entre el pin 3 del módulo arrancador al pin D2 del Arduino Nano, las fuentes de alimentación del pin 2 del módulo arrancador al GND del Arduino Nano (pin 3) y entre el pin 1 del módulo arrancador al pin 1 del resistor y a VCC del Arduino Nano (pin 26). Esto último resulta del VCC del Arduino Nano que tiene una salida de 5V el cual es encargado de alimentar a cada uno de los componentes por lo que se tiene que realizar la conexión entre el VCC del Arduino Nano a la resistencia y al módulo arrancador.

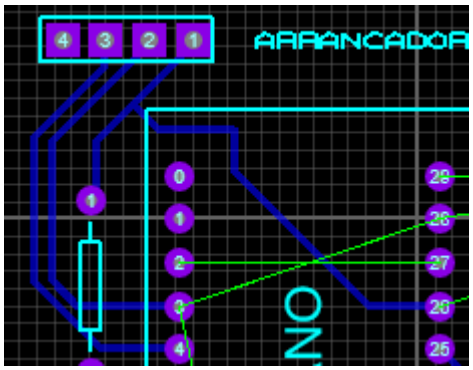


Imagen 144. Conexiones módulo arrancador al Arduino Nano.

De la misma resistencia se debe tener conexión al pulsador, entonces el pin 2 del resistor se debe conectar al pin 1 del pulsador.

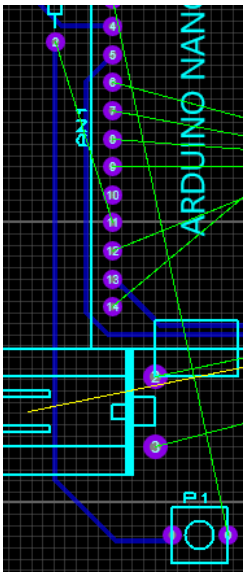


Imagen 145. Conexión entre el resistor y el pulsador.

Continuando con las demás conexiones entre el Arduino Nano y el puente H se obtiene lo siguiente:

Pin 10 del puente H al pin D7 (pin 9)

Pin 11 del puente H al pin D6 (pin 8)

Pin 12 del puente H libre por el momento

Pin 13 del puente H al pin D5 (pin 7)

Pin 14 del puente H al pin D4 (pin 6)

Se recorrió la línea del resistor al pulsador, para que la vía pase por debajo del motor izquierdo, tal como se muestra.

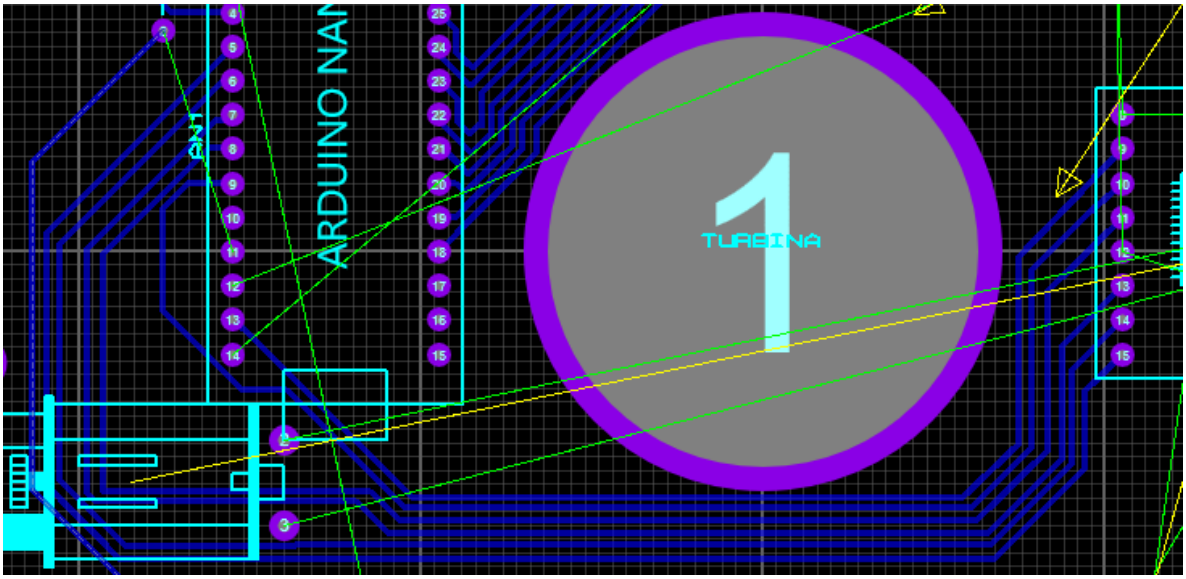


Imagen 146. Conexiones entre el Arduino Nano y puente H.

Las siguientes serán conexiones de alimentación. De principio se conectan ambos Switch por medio del pin 2 y del pin 2 del Switch izquierdo se realiza una conexión al pin 2 (pin positivo) del

conector para la batería. Después se conecta el pin 0 del conector de la batería al pin 2 del puente H. En la parte superior se conecta el pin 1 (VCC) del ESC al pin 9 del conector para los sensores y del Switch izquierdo se conecta el pin 3 al pin 2 (Positivo) del VIN T, el pin 0 del ESC se conecta al pin 1 del VIN T, del pin 29 del Arduino Nano al pin 3 del Switch 1.

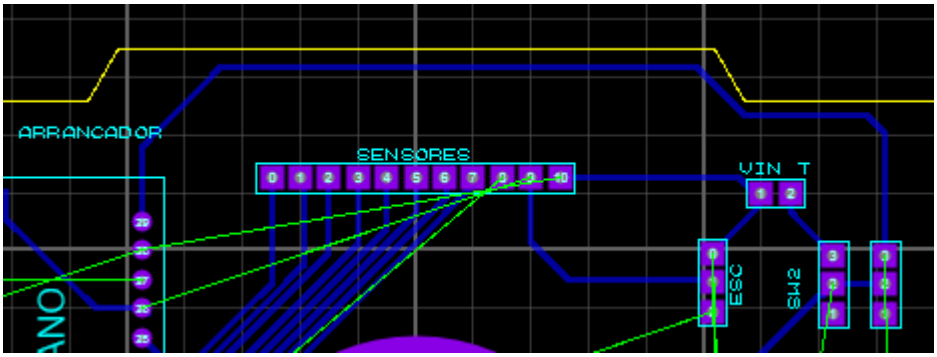


Imagen 147. Conexiones de alimentación superior.

Las conexiones entre los motores son las siguientes: para el motor derecho no existe problema puesto que se pueden conectar de forma directa con el puente H, conectando el pin 3 del motor al pin 6 del puente H y el pin 2 del motor al pin 5 del puente H.

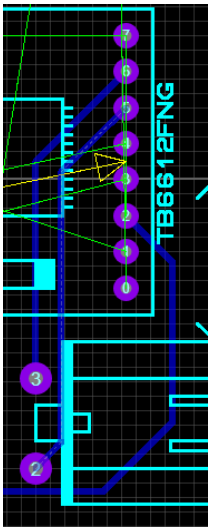


Imagen 148. Conexiones del motor derecho al puente H.

Para el motor izquierdo por cuestión de espacio en la placa se deben realizar unas perforaciones, entonces seleccionando el pin 5 del puente H, se inicia el recorrido normal de la vía, pero justo cuando se encuentre en punto de colisión con otra, con doble click la línea en automático pasa de azul a rojo, lo que indica que la conexión ya no es inferior sino superior, posterior a esto cuando la vía este cerca del Pad 2 del motor izquierdo nuevamente con doble click la vía vuelve a ser inferior (azul) y la conexión se termina sin problema alguno. Mismo caso para el pin 4 del puente H que termina en conexión al Pad 3 del motor izquierdo.

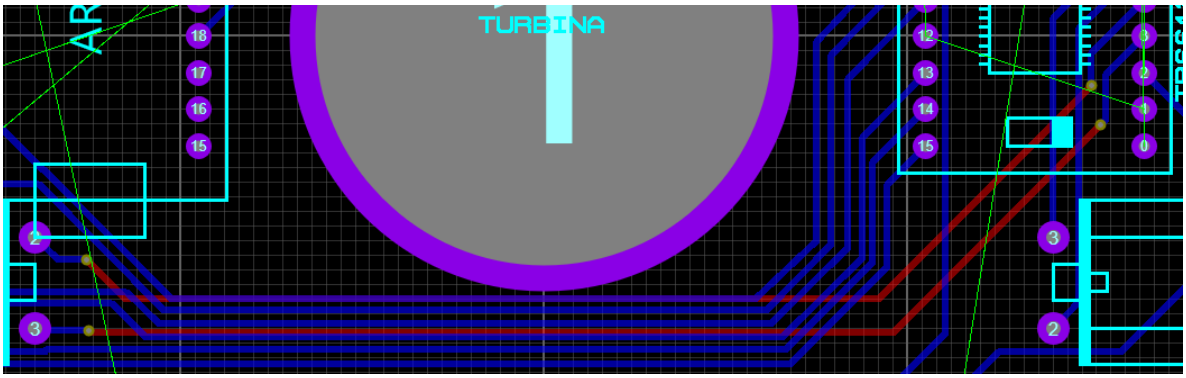


Imagen 149. Conexiones motor izquierdo a puente H.

La conexión de alimentación para la turbina sale del pin 2 del ESC y llega al pin 12 del Arduino Nano.

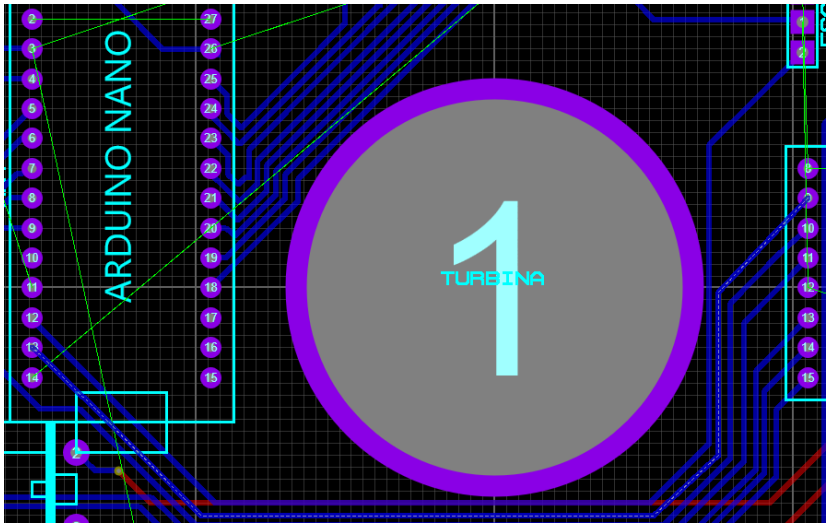


Imagen 150. Conexión de la turbina al Arduino Nano.

Es importante cambiar el grosor de las vías de conexión a 50 milipulgadas para que la corriente pueda fluir de manera normal y sin problema alguno, para esto, con click derecho sobre alguna vía de alimentación se debe seleccionar la opción “Change Trace Style” y T50.

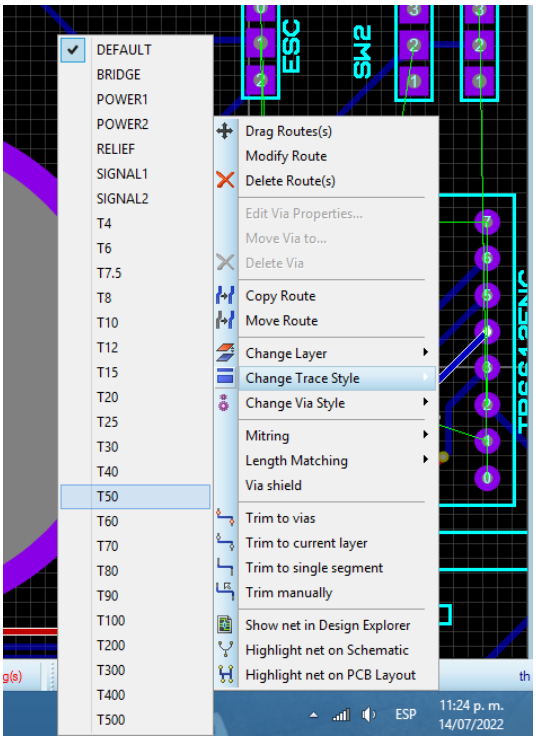


Imagen 151. Cambio de grosor de las vías.

Una vez realizado el cambio del grosor de las vías, también se debe cambiar el grosor de los Pads de perforación para las líneas que cambian de color azul a rojo, esto fue por una placa de una sola cara utilizada en el proyecto. Las perforaciones sirven para evitar que las vías pasen unas sobre otras, y que de forma externa se pueda realizar la conexión en la parte superior por medio de algún cable. Esto puede ser innecesario con una placa de doble cara. Para realizar el cambio de Pads, al igual que con las vías, con click derecho sobre los Pads amarillos se debe seleccionar la opción “Change Via Style” y elegir la opción V70X30.

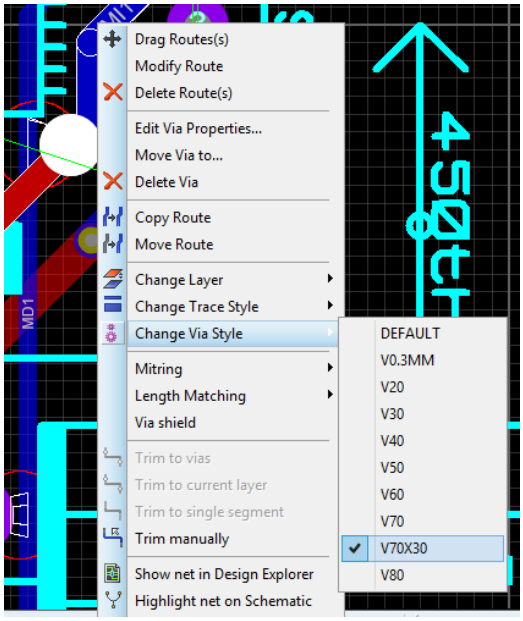


Imagen 152. Cambio de grosor de los Pads de unión.

Cuando se cambia el valor de los Pads estos aumentan su tamaño y por ende puede suceder un error de invasión de espacio, en este caso generado por un Pad con una vía, y por tanto el programa marcar el error con un círculo tal como se aprecia en la imagen siguiente:

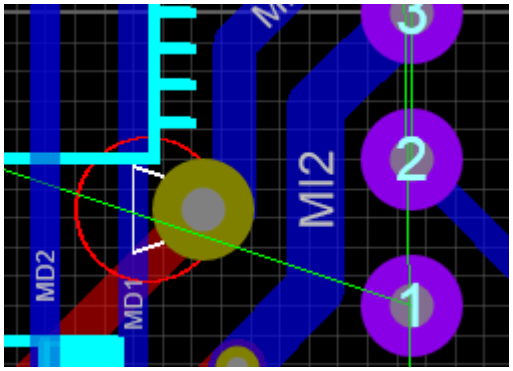


Imagen 153. Colisión entre pad y vía.

Para resolver este problema de colisión lo único que se debe hacer es mover la vía o el Pad marcados, con ello el error se resuelve, el único problema es que con muchas vías en el medio al mover una o un Pad este puede chocar involuntariamente con otro.

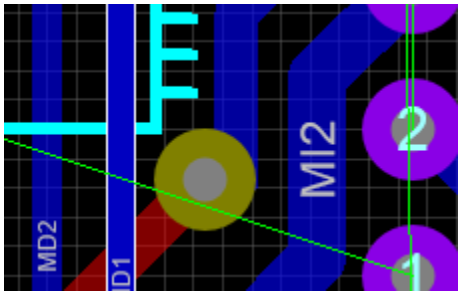


Imagen 154. Colisión entre pad y vía resuelta.

Se debe hacer otro puente entre la barra de sensores y el Arduino Nano, esta conexión se realizara del pin 8 del conector de sensores, al pin 14 del Arduino Nano, en el que se utilizarán 2 puentes para la conexión.

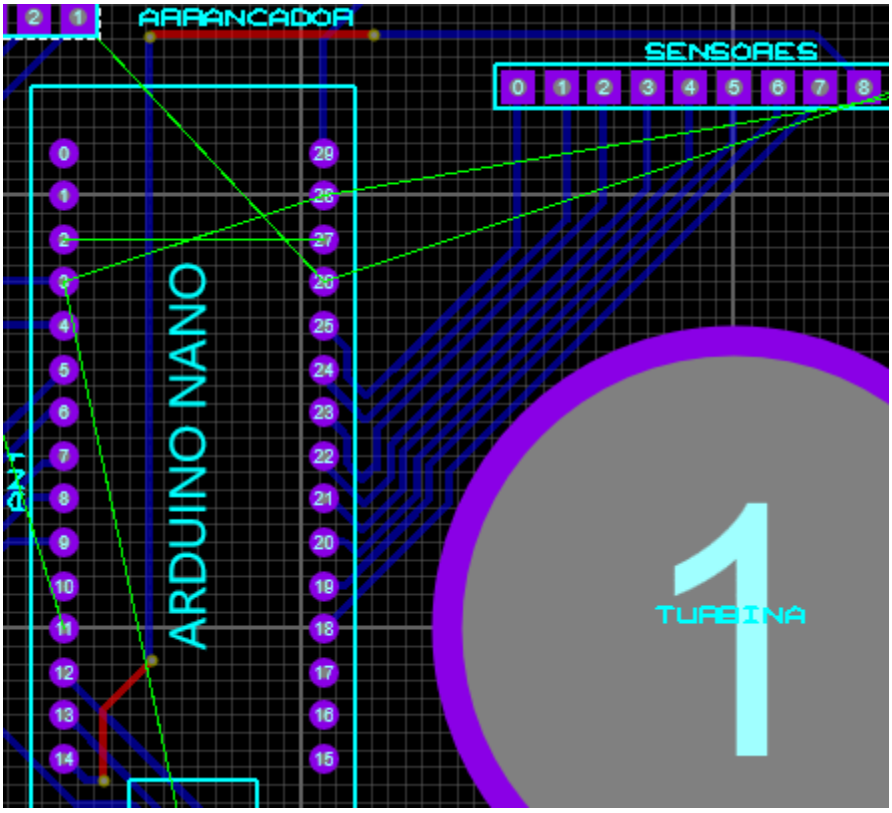


Imagen 155. Conexión Arduino Nano a barra de sensores.

La conexión para la información del pulsador que debe llegar al Arduino Nano, fue realizada en un principio entre el pulsador al pin 2 del resistor, ahora lo que hay que realizar es la conexión entre ese pin con el pin 11 del Arduino Nano, para esto la conexión de la vía debe ser por la parte superior pasando por arriba del arrancador, para descender por la parte del Arduino Nano llegando al pin 11.

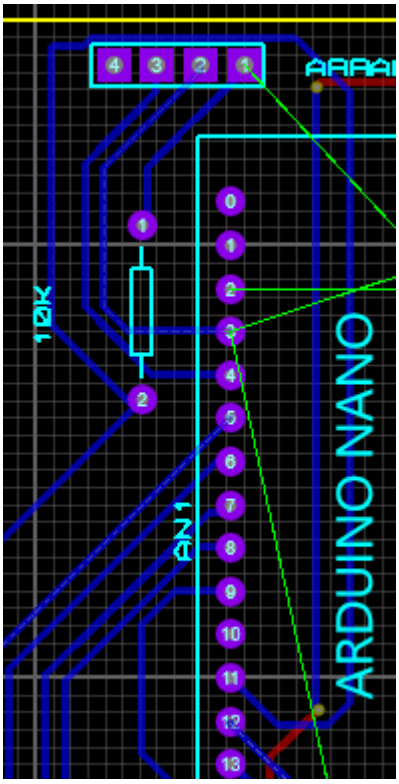


Imagen 156. Del pulsador, al resistor, y del resistor al Arduino Nano.

Como la conexión entre el Pad 1 del arrancador con el Arduino Nano fue eliminada, debe hacerse de nuevo, pero ahora implementando un puente el cual va a conectar al pin 1 del arrancador, con la vía del pin 1 del resistor con el pin 26 del Arduino Nano.

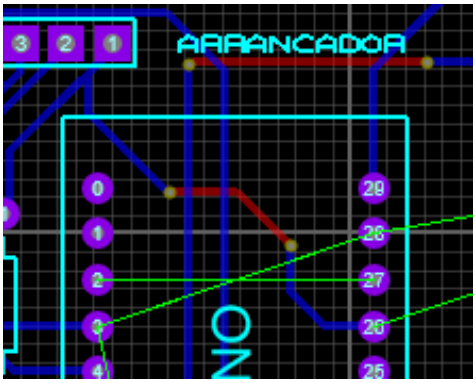


Imagen 157. Del arrancador, al resistor, al Arduino Nano.

Adicional a esta conexión que termina en el pin 26 del Arduino Nano, también se debe sacar una conexión al conector de la barra de sensores de ese mismo pin, justamente al pin 9 del conector de la barra de sensores, pasando la vía por un costado del Arduino Nano, y por encima de dicho conector.

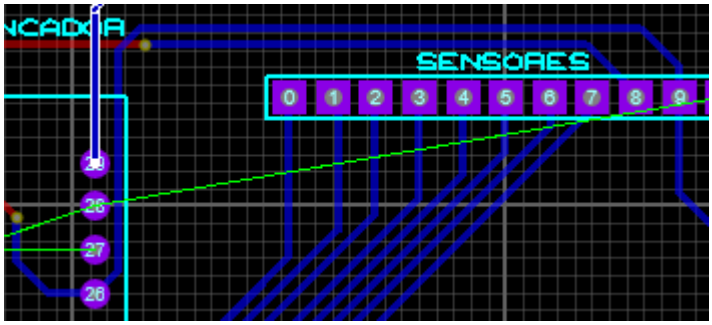


Imagen 158. Del arduino nano, al conector de la barra de sensores.

Ahora bien, dentro del Arduino Nano existen un par de conexiones entre GND que no necesitan ser conectadas, aunque así lo marca el programa, estas conexiones son entre los pines 2 y 28 del Arduino Nano, únicamente se omiten. El pin 28 del Arduino Nano si será conectado, de igual forma con el conector de la barra de sensores, directo al pin 10 del conector. Pasando por el lado izquierdo del Arduino Nano por dentro, hacia la parte superior, utilizando un puente para llegar por la parte superior al pin 10 del conector.

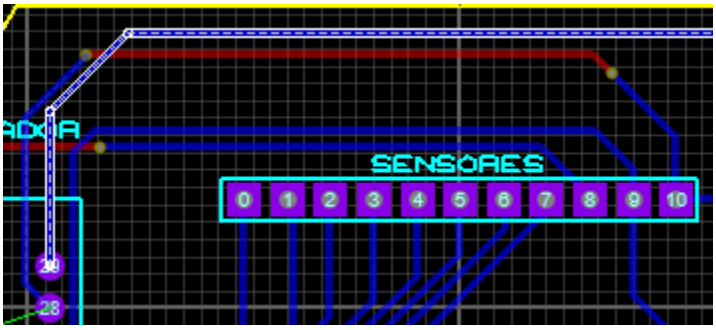


Imagen 159. Del Arduino Nano, al conector de la barra de sensores.

El voltaje de entrada será la conexión del pin 0 del puente H y el pin 3 del Switch de la derecha, apoyados de un puente que debe ser utilizado en base a las vías colocadas, para este caso el ejemplo es el siguiente:

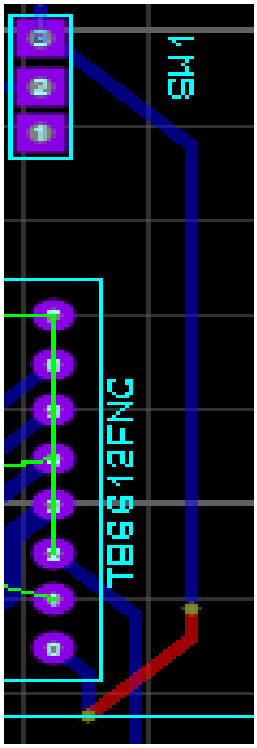


Imagen 160. Switch con puente H.

Ahora queda por hacer otra conexión entre el puente H con el ESC, justamente el pin 0 del ESC debe ir conectado al pin 8 del puente H, y también con ayuda de un puente se realizara la conexión.

Nuevamente es importante cambiar el grosor de las vías cuando las vías son para alimentación, como este caso, posteriormente se presentara un esquema general que muestre todas las vías de conexiones y cuales necesitan un cambio de grosor, porque no son todas.

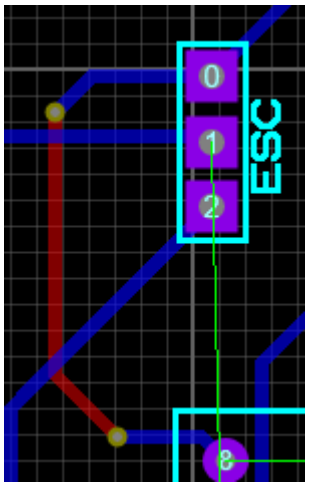


Imagen 161. ESC con puente H.

Resta por realizar otra conexión entre el ESC y el puente H, que será entre el pin 1 del ESC con el pin 12 del puente H, también una conexión entre dos pines del puente H, que son el pin 7 y 8 con el pin 0 del ESC. Y por último con el puente H también se debe realizar una conexión entre el pin 1 con el pin 1 del ESC o a una vía que conecte con dicho pin del ESC.

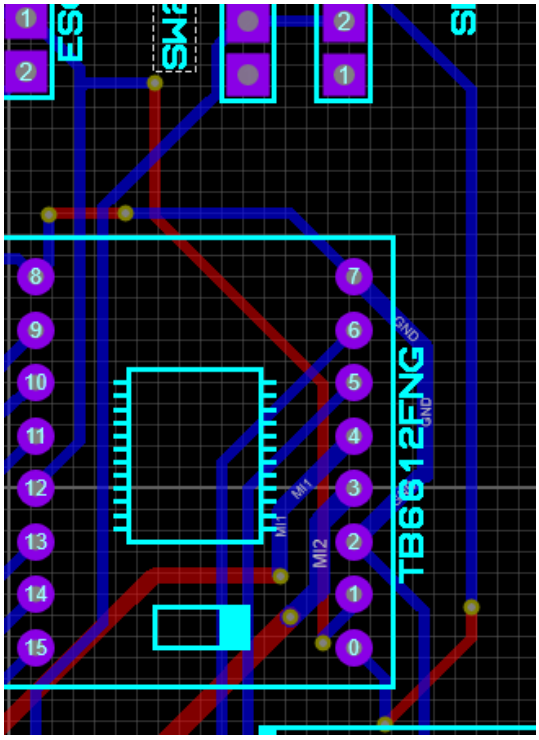


Imagen 162. ESC con puente H y entre pines del puente H.

Por último se debe realizar una conexión entre el pin 0 del pulsador, con el pin 0 del conector de la batería y ese mismo pin 0 con el pin 2 del puente H.

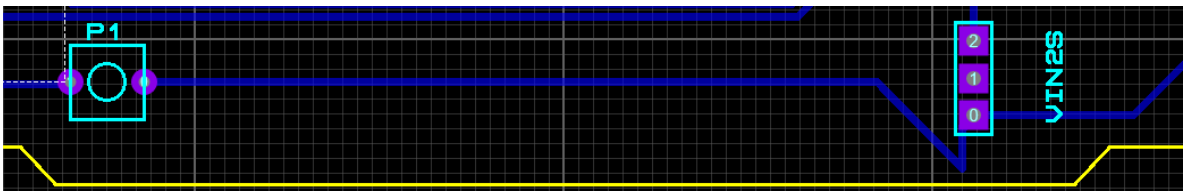


Imagen 163. Pulsador con fuente de alimentación de la batería.

Como dato importante, en una vista general de las conexiones de vías de toda la placa es importante verificar que no existan líneas de color verde por la placa, si este es el caso, quiere decir que las conexiones están completas y no hace falta ninguna conexión entre dispositivos y pines, de lo contrario si se marcan líneas verdes se tendría que hacer una revisión de la conexión que hace falta, entre que dispositivos o pines y realizar la conexión correspondiente. Existe una excepción

la cual es para 4 pines del Arduino Nano, como esos pines tienen las mismas salidas no aplica conexión de vía alguna, pero es tan solo para estos 4 pines, y nada más. La siguiente es una vista general de cómo se trabajaron las conexiones de vía.

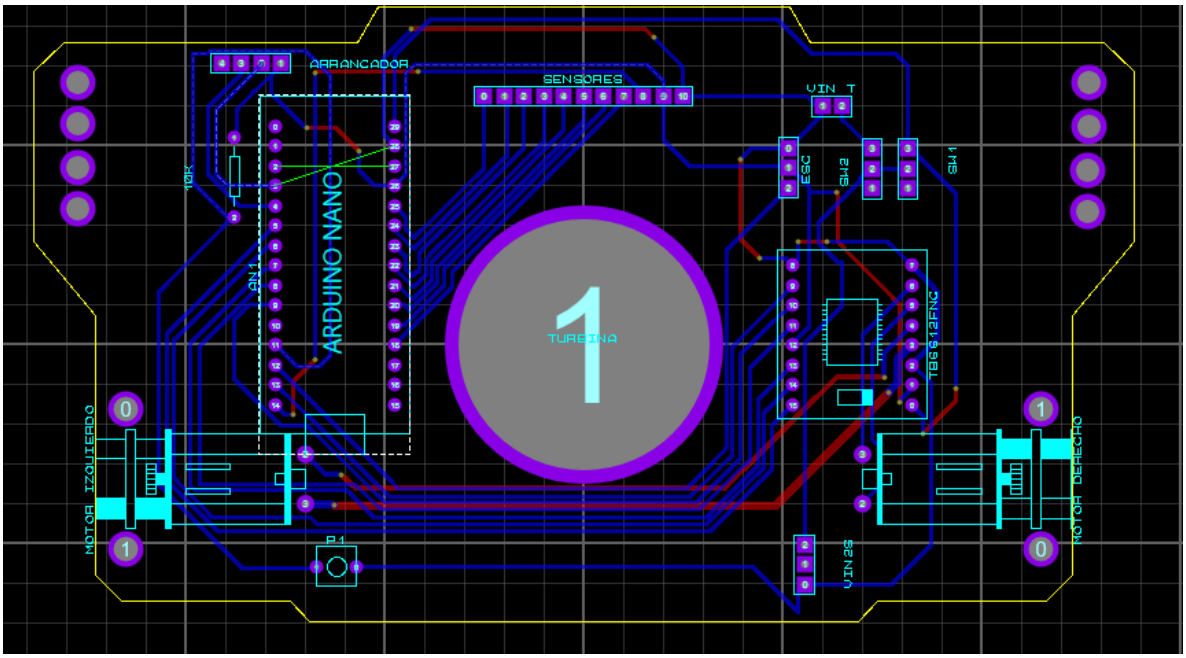


Imagen 164. Conexiones de vías para la placa.

Aquí se muestran las conexiones realizadas, pero con un tamaño de vías general. En una siguiente imagen se podrá observar un cambio en el tamaño de las vías, principalmente las de alimentación, pero además de ello se podrán observar otras perforaciones en la placa que como se ha mencionado previamente no son necesarias en su totalidad, sin embargo para el desarrollo del proyecto sí fueron colocadas, no obstante, para cualquier caso se pueden colocar y lo único a realizar son los mismos pasos tal cual se ha trabajado.

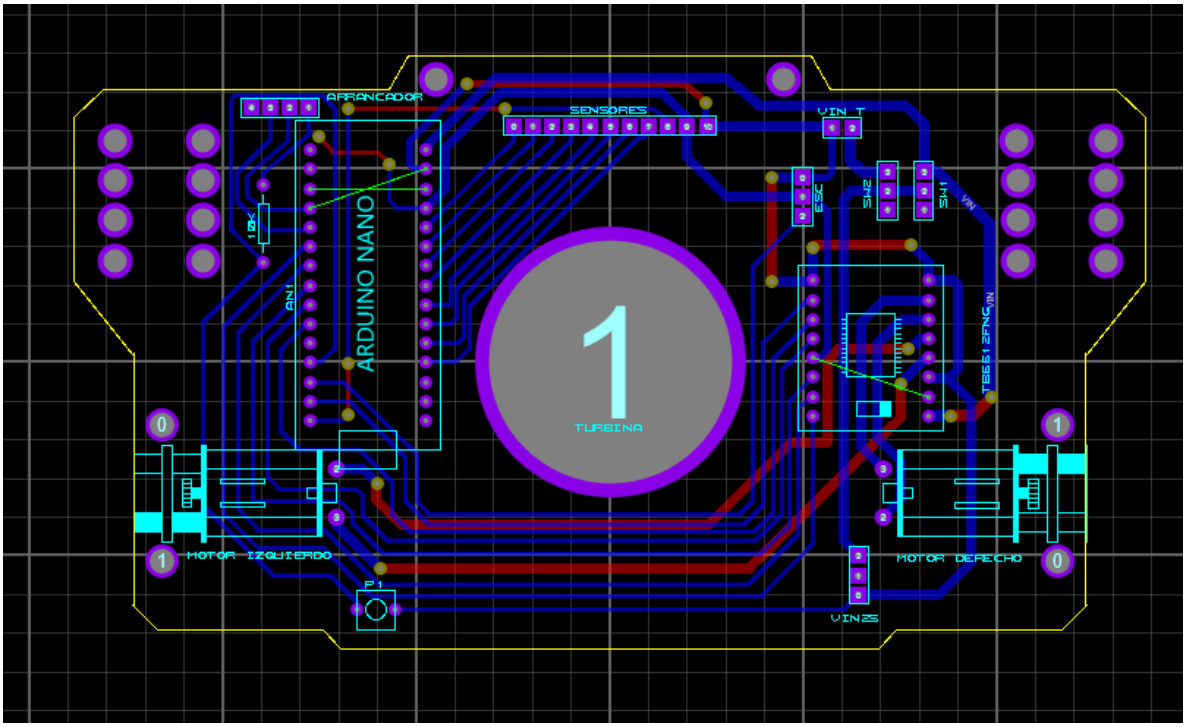


Imagen 165. Conexiones de vías para la placa Exia.

Para la placa del proyecto la cual se ha denominado “Exia” se colocaron dos hileras de perforaciones en la parte superior a ambos costados con el fin de utilizar tanto una barra de 16 sensores multiplexada como de 13 sensores, en la parte central por encima del conector de la barra de sensores se colocaron otras dos perforaciones las cuales sirven para una barra de 8 sensores que bien puede ser una QTR8 de Pololu. Con eso se han realizado las perforaciones más importantes respecto a las barras de sensores, en lo cual nuevamente se reitera el ahorro de estas perforaciones sabiendo que tipo de barra de sensores se va a utilizar.

Pasando a otro tema importante, el espacio en el caso de la placa original fue planeado para realizarse de forma artesanal con baquelita, pero al no tener experiencia trabajando con este tipo de material y su proceso de elaboración, esto puede resultar difícil y tardado, pero no imposible y en caso de trabajarlo así, esta puede ser una opción más económica que realizar la placa de manera

profesional por una empresa. En este caso como la idea principal era trabajar la placa de esta manera la única opción era obtener la placa a una sola cara, y es aquí donde entra la parte del espacio, siendo únicamente una placa de una cara se tiene un espacio limitado para realizar las vías, es por ello que se utilizan puentes para realizar conexiones y que pasen por encima de otras.

Por falta de experiencia en placas artesanales las placas realizadas para el proyecto fueron enviadas a una empresa profesional dedicada a hacer todo tipo de placas, pero aquí también es importante mencionar que dichas empresas solicitan exportar la placa en un formato especial tipo Gerber, para que ellos puedan trabajar con el diseño e imprimirlas. Este tema del formato será abordado en breve.

Como siguiente, es importante realizar una revisión de la placa, pero ahora pasando de un 2D a un 3D, con ayuda del 3D Visualizer se puede observar el diseño de la placa, el orden de los componentes, las perforaciones, y las conexiones de vía tanto inferiores como superiores.

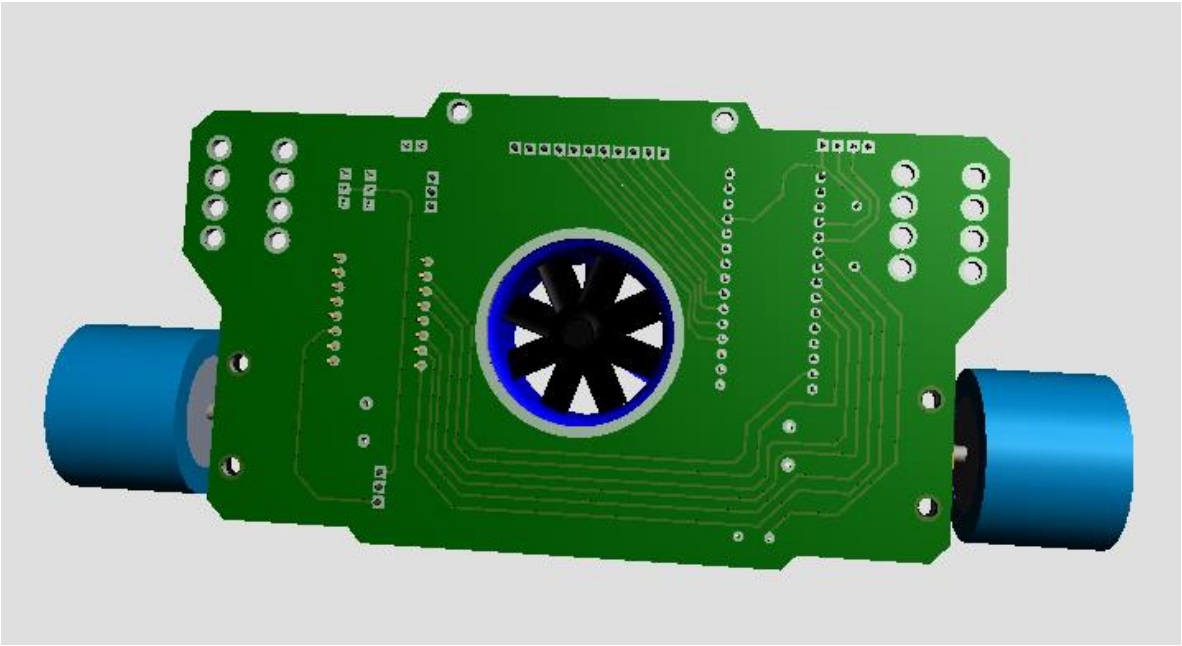


Imagen 166. Vista inferiores de conexiones de vía.

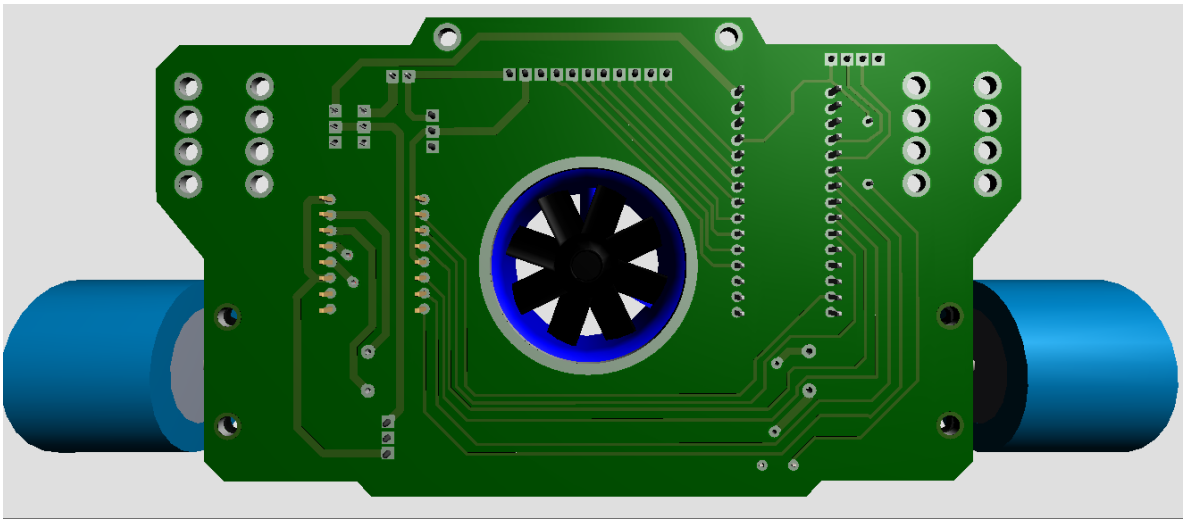


Imagen 167. Vista inferiores de conexiones de vía con aumento de tamaño para vías de alimentación.

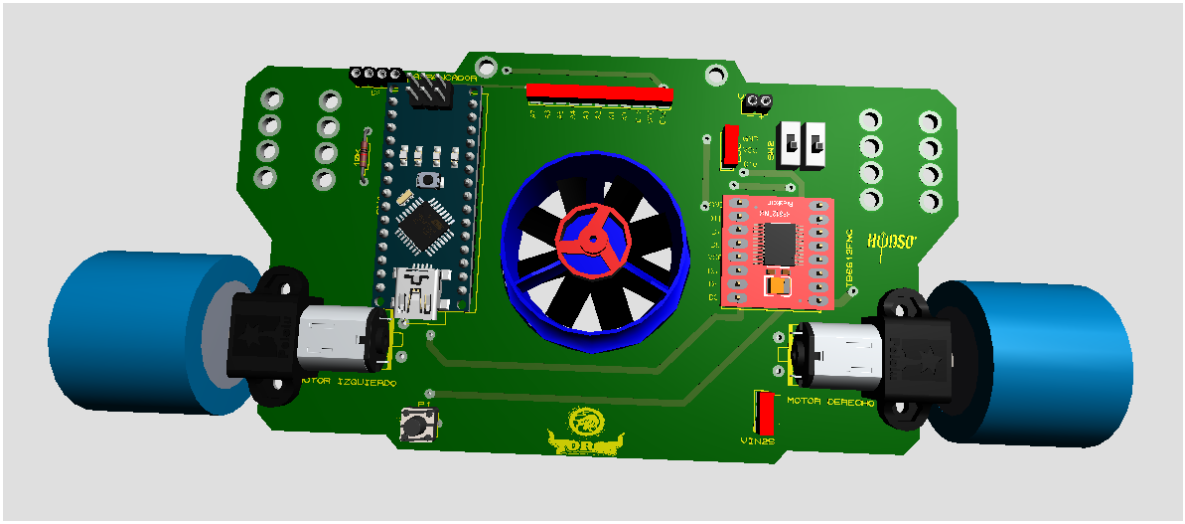


Imagen 168. Vista superior de conexiones de vía con aumento de tamaño para vías de alimentación.

4.5.4 Revision final de la placa (pcb)

Concluidas todas las conexiones de vía y todas las perforaciones, se debe realizar una última revisión final de la placa, en la parte inferior de Proteus se puede comprobar que no hay errores con una palomita verde y la leyenda “No DRC Erros” para al fin continuar con la fase final de la placa.

En la parte superior de Proteus se encuentra la opción “Tools” y dentro de ella se debe seleccionar la opción “Power Plane Generator”

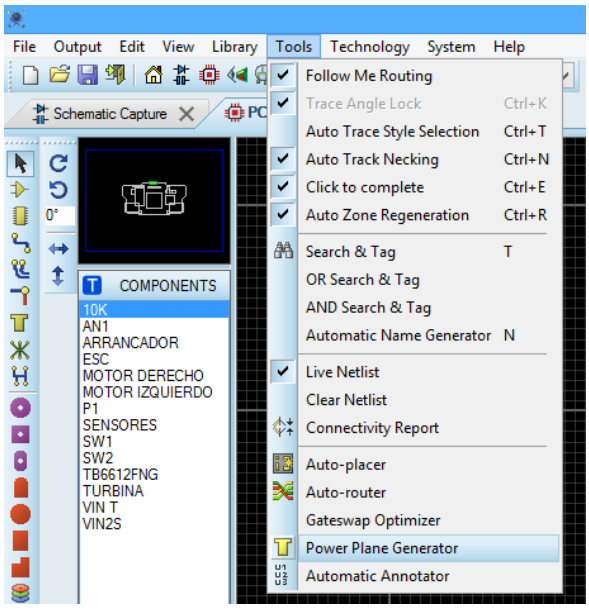


Imagen 169. Power Plane Generator.

Como resultado de esa opción se genera una base para la placa, posterior a la selección del “Power Plane Generator” se mostrará una ventana propia de la opción seleccionada la cual simplemente debe ser aceptada dando click en “OK”

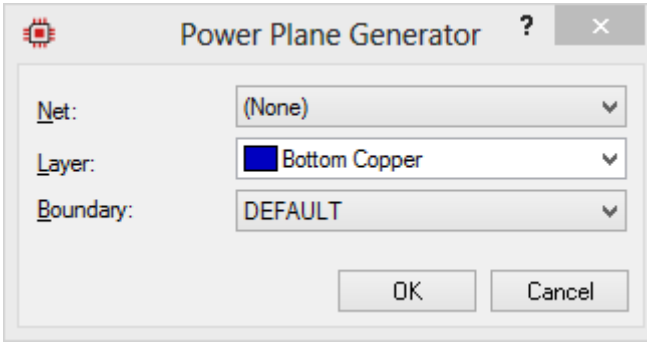


Imagen 170. Power Plane Generator (ventana).

Inmediatamente a la acción se pueden observar cambios en la placa de forma general. La placa pasa de tener el fondo negro del espacio de trabajo a un color azul rey tal cual lo muestra la venta del “Power Plane Generator”

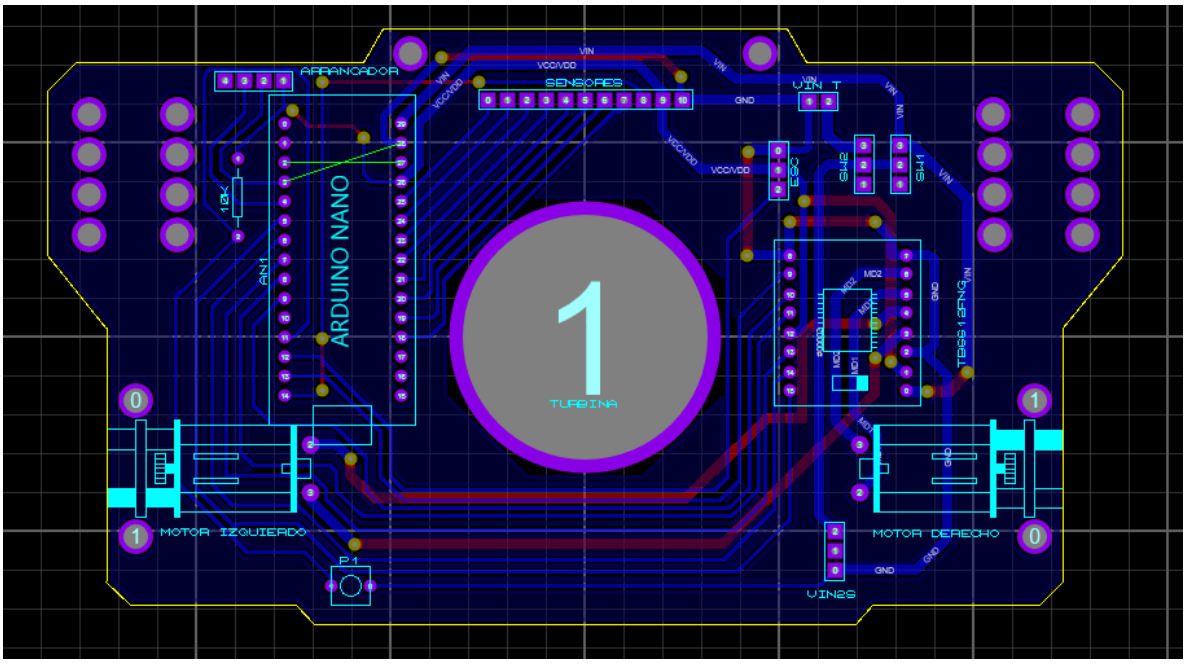


Imagen 171. Placa generada.

Con la placa generada lo siguiente es modificar sus propiedades, para editar las propiedades se debe aplicar doble click izquierdo sobre alguna área de la placa que no tenga vías o Pads de perforación, esta acción arrojara una ventana de edición para la placa.

Las características a modificar serán: justo a un costado de BOTTOM COPPER se debe hacer el cambio de Dimmed por Normal, en Boundary se debe cambiar DEFAULT a RELIEF, también modificando la separación en CLEARANCE cambiando el valor de 0 a 40 milipulgadas. Siendo esas las únicas dos modificaciones para las propiedades de la placa.

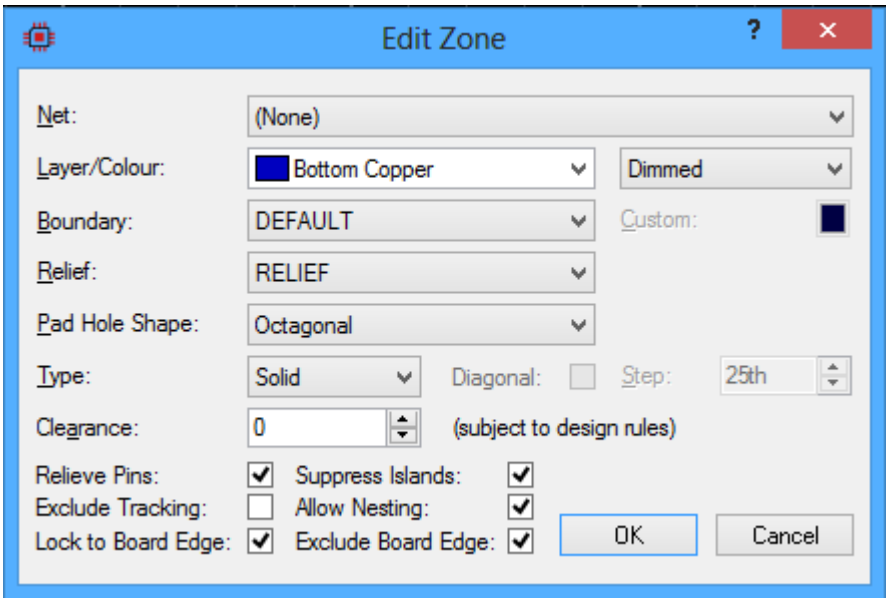


Imagen 172. Edición de propiedades de la placa.

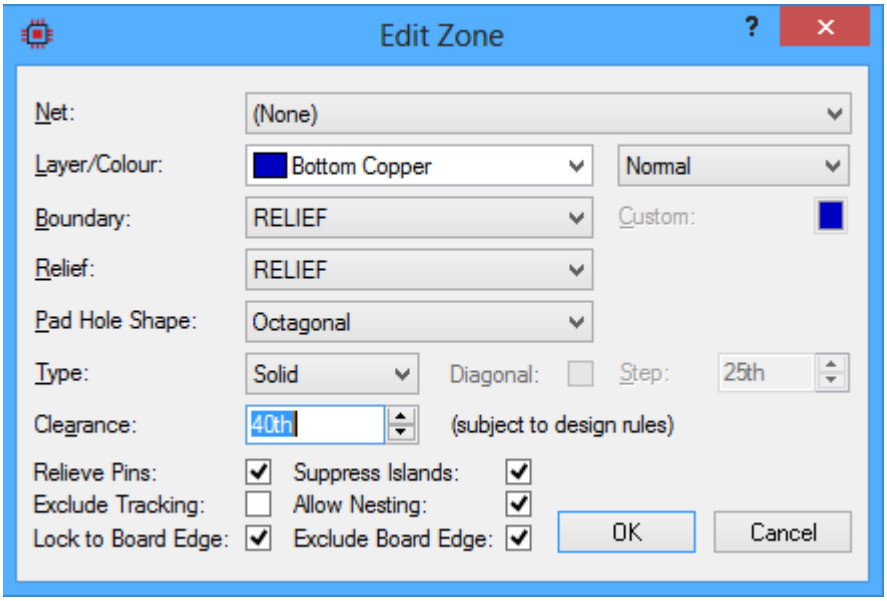


Imagen 173. Propiedades de la placa editadas.

Una vez realizados estos cambios se puede observar a la placa final de forma general, que en realidad no ha tenido cambios relevantes en cuanto a lo realizado en cuestión de software pero que

en cuestión física si se pueden apreciar muchas más. Las siguientes son vistas de la placa final en 2D y 3D.

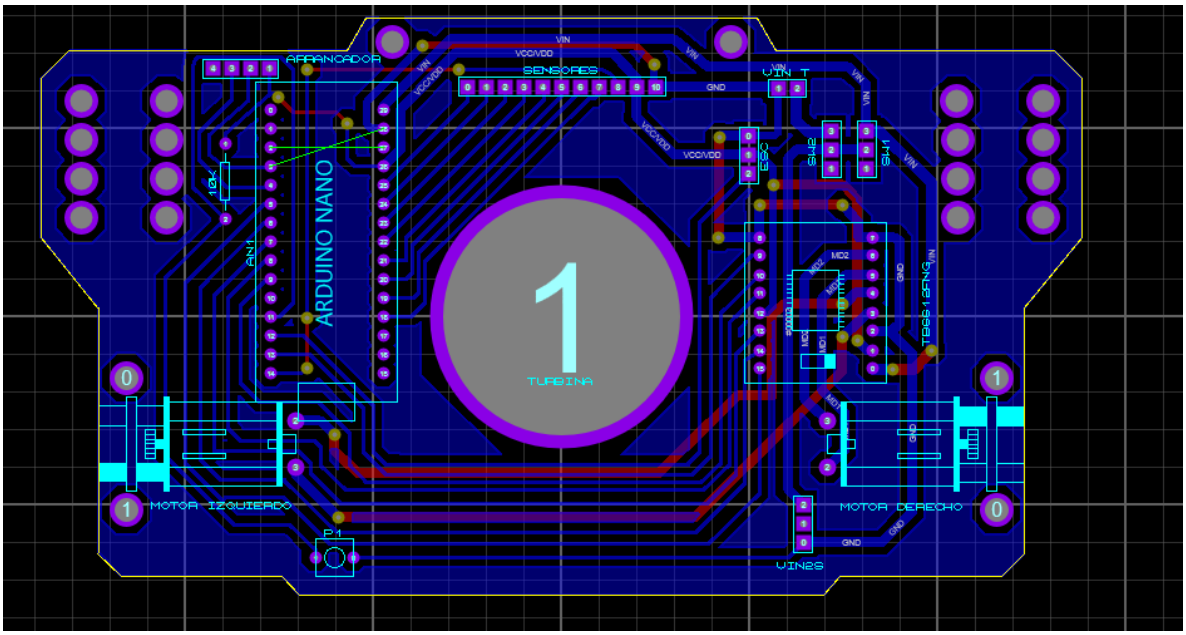


Imagen 174. Placa final 2D.

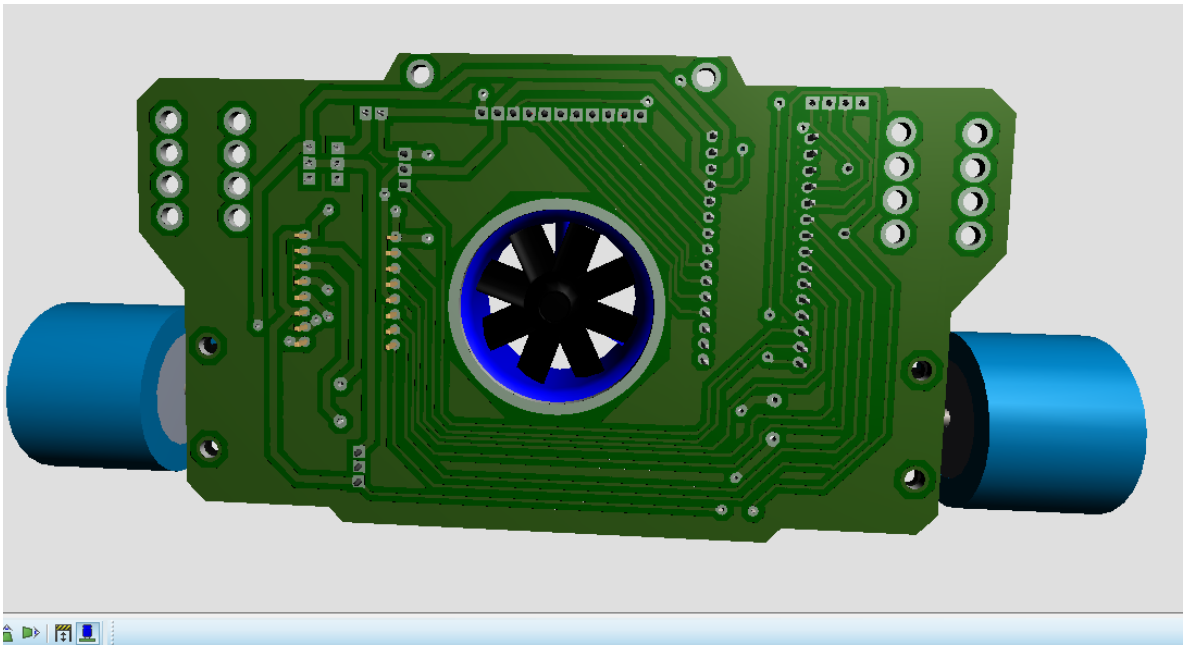


Imagen 175. Placa final 3D

4.5.5 Nombre de las conexiones

Es importante tener una referencia de cómo se colocaron los dispositivos para la placa física, por tal motivo se tiene que colocar el nombre a las conexiones de los dispositivos para saber el orden de cómo fueron conectados, para que al momento de realizar conexiones y enviar corriente por el dispositivo no exista ningún problema, por este motivo se colocan nombres o símbolos que faciliten esta labor.

Arrancador

Para el arrancador las conexiones son las siguientes: El pin 4 libre, el pin 3 D2, al pin 2 negativo o GND (como se entienda mejor), y para el pin 1 positivo o VCC.

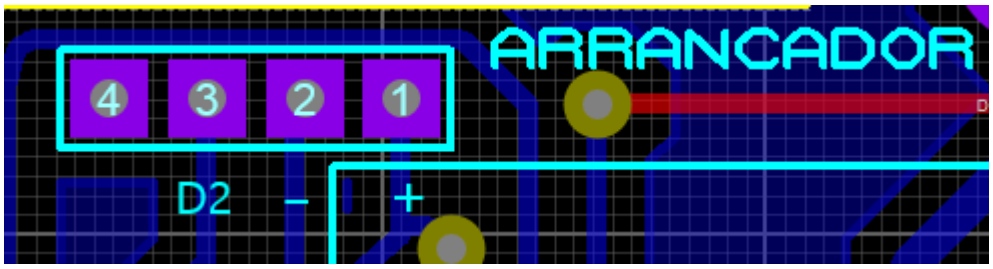


Imagen 176. Símbolos para el arrancador.

Conector para la barra de sensores

Para la barra de sensores las conexiones son las siguientes: El pin 0 A7, el pin 1 A6, el pin 2 A5, el pin 3 A4, el pin 4 A3, el pin 5 A2, el pin 6 A1, el pin 7 A0, el pin 8 D12, el pin 9 VCC o positivo y el pin 10 GND o tierra.

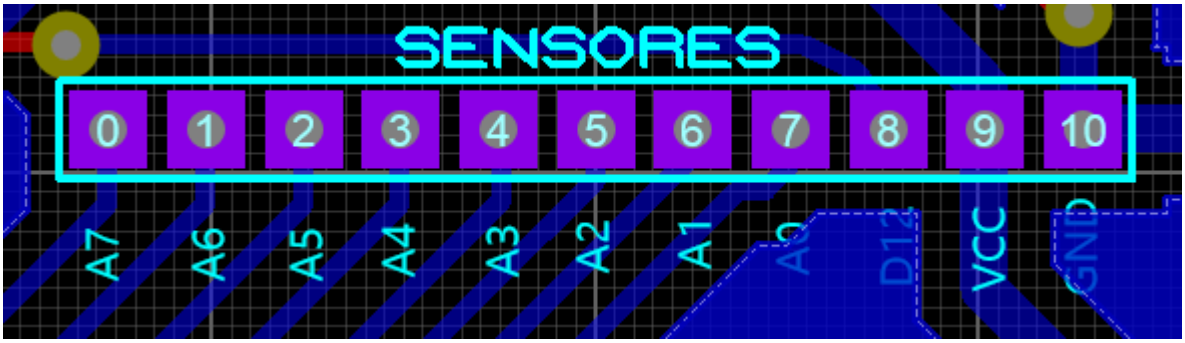


Imagen 177. Símbolos para el conector para la barra de sensores.

Conector para ESC

La conexión para el ESC es la siguiente: El pin 0 GND, el pin 1 VCC y el pin 2 D10.

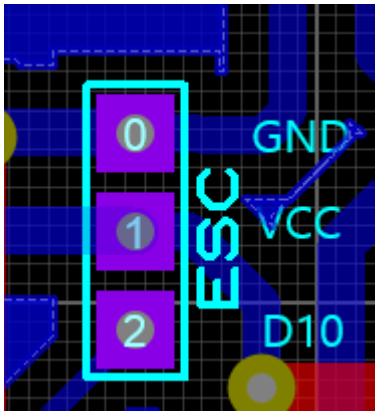


Imagen 178. Símbolos para el conector para la barra de sensores.

Conector VIN T para ESC

Las conexiones para el VIN T son las siguientes: El pin 1 positivo o VCC, y pin 2 negativo o tierra.

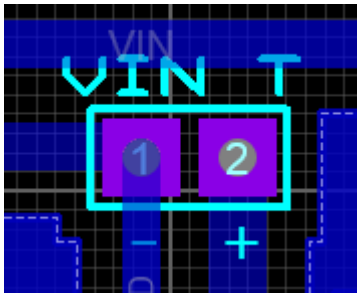


Imagen 179. Símbolos para el conector VIN T para el ESC.

Conexiones de vías para el puente H

Las conexiones de vías que salen del puente H también deben tener un nombre para poder ser referenciados, en este caso el orden es el siguiente: El pin 8 GND, el pin 9 D11, el pin 10 D7, el pin 11 D6, el pin 12 VCC, el pin 13 D5, el pin 14 D4, el pin 15 D3.

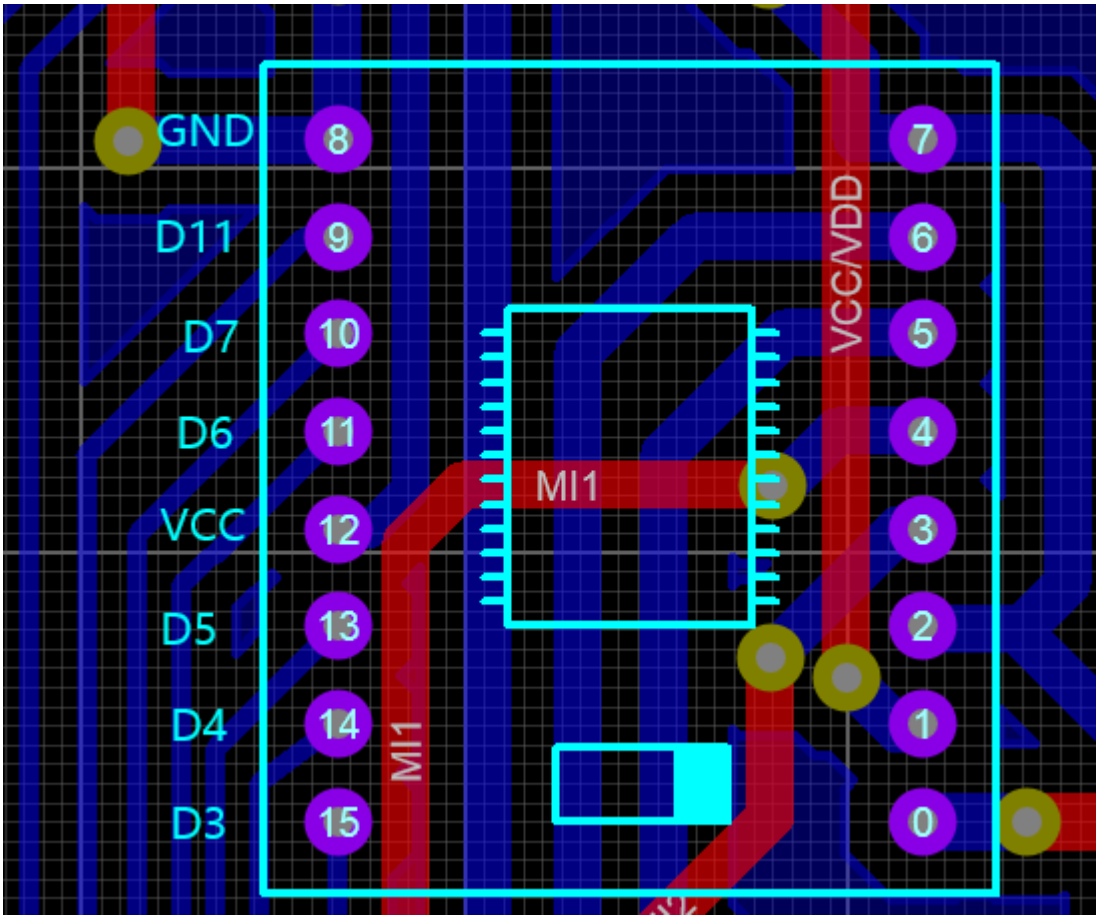


Imagen 180. Símbolos para el puente H.

Conexión de la batería

La conexión para la batería es la siguiente: Para el pin 2 positivo, para el pin 1 libre y para el pin 0 negativo.

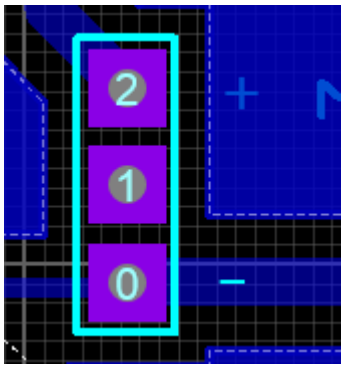


Imagen 181. Símbolos para la batería.

4.5.6 Logotipos

Los logotipos son una parte importante de cualquier proyecto, ya sea escolar o empresarial, para este caso siendo un proyecto escolar es importante tener logotipos que representen tanto a la institución como al club de robótica (o cualquier club) que sea parte del proyecto, por este motivo se colocó un logotipo de “Toros TEC” y un par de logotipos personales.

Para colocar los logotipos es necesario seguir con los pasos a continuación:

1. Lo primero es tener un logotipo, ya sea personal, institucional, empresarial, etc. Ese logotipo debe ser una imagen guardada en alguna carpeta específica del ordenador.

2. La imagen debe abrirse con Paint para poder guardarla desde este programa, para ello en la parte superior de Paint se debe seleccionar “Archivo”>”Guardar como” y elegir la opción “Imagen BMP” o Mapa de Bits.

Imagen 182. Imagen en mapa de Bits.

3. La imagen en Mapa de Bits preferentemente debe ser guardada en la misma carpeta que la imagen original.

4. Al momento de guardar la imagen se debe seleccionar la opción “Mapa de bits monocromático”.

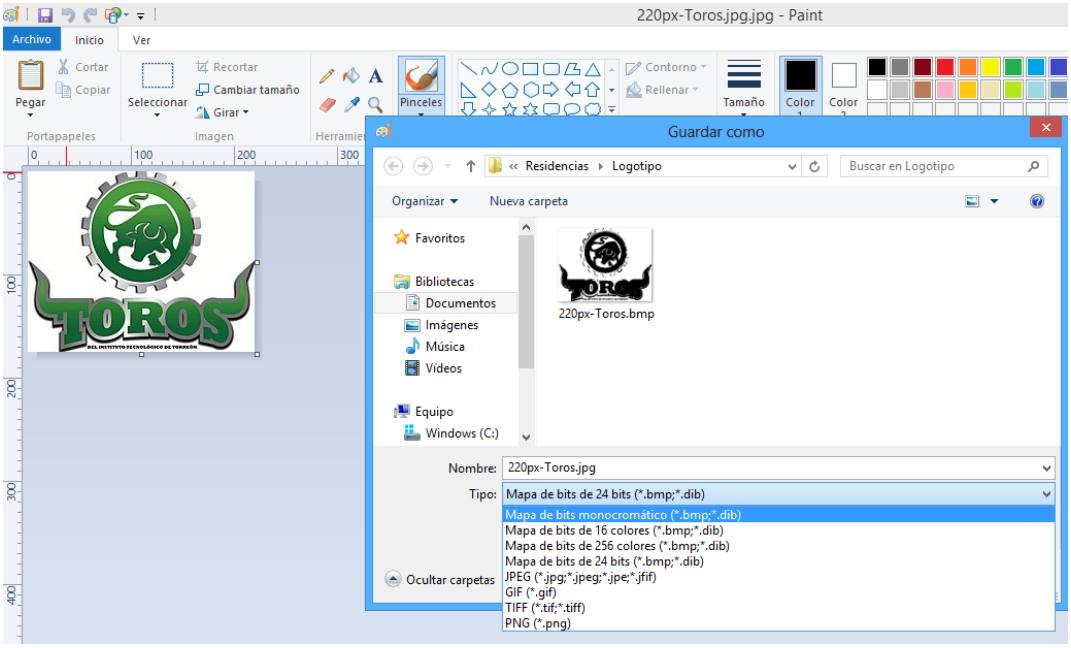


Imagen 183. Imagen en mapa de Bits monocromático.

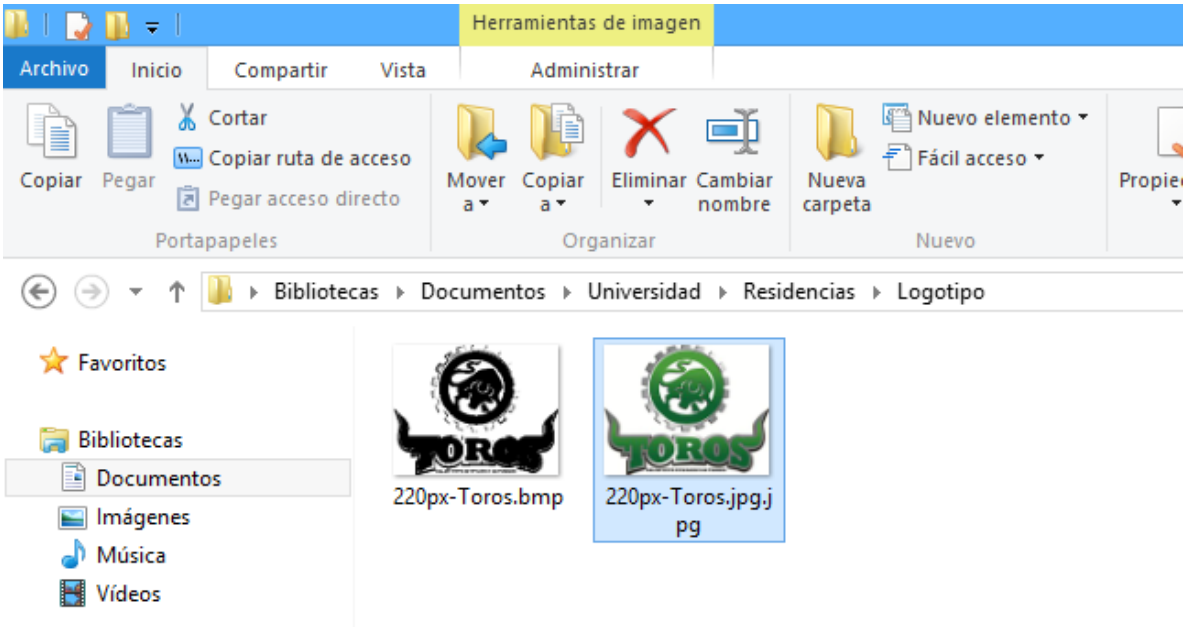


Imagen 184. Archivo en mapa de Bits monocromático en carpeta.

5. Lo siguiente es ir a Proteus y exportar el archivo. En la parte superior izquierda del programa se encuentra la opción “File” ahí se debe elegir la opción “Import Bitmap” para buscar el logotipo dentro de la carpeta.

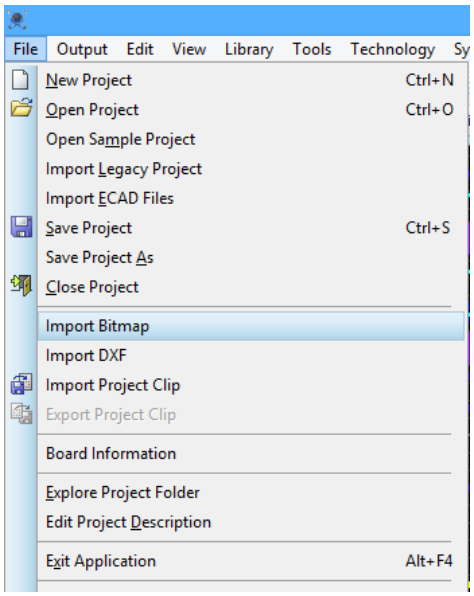


Imagen 185. Importando mapa de Bits.

6. Se debe insertar el logotipo y dependiendo de las dimensiones del archivo serán las dimensiones del logotipo, lo que resta es reducir o aumentar su tamaño según sean necesario.

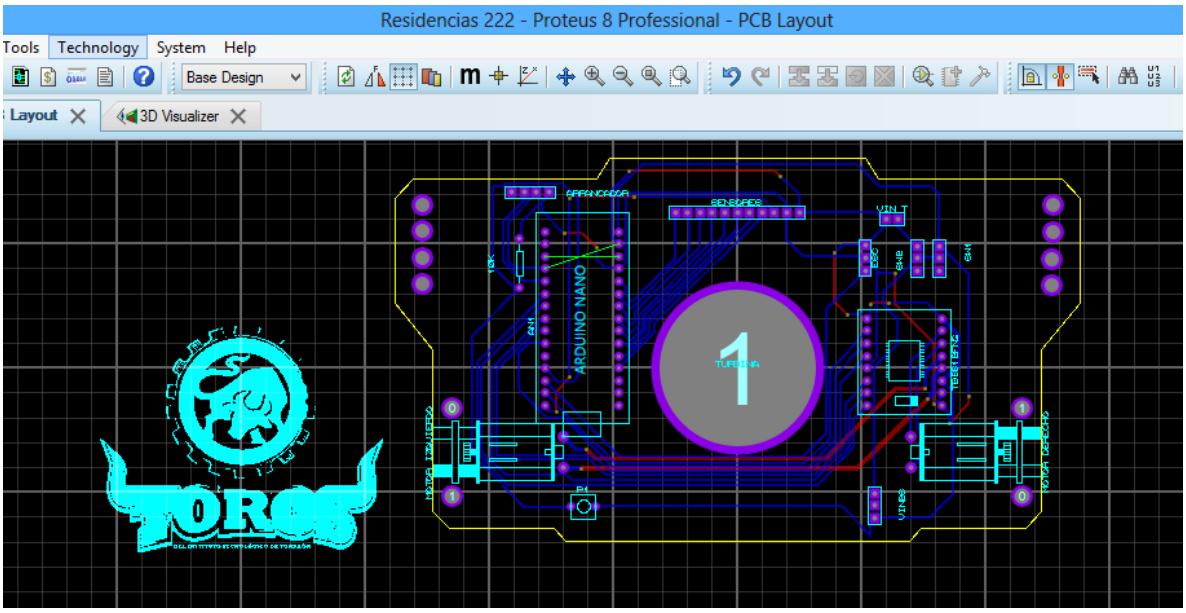


Imagen 186. Importando logotipo.

7. Lo que resta por hacer únicamente es colocar el logotipo en la superficie de la placa con la ubicación deseada.

8. Para que el logotipo se muestre en la parte inferior de la placa lo que se debe hacer es copiar el logotipo, click derecho, utilizar el efecto espejo en X, dar nuevamente click derecho sobre el logotipo y seleccionar “Change Layer” y cambiar “Bottom Silk” a “Top Silk” para poder ubicar un logotipo en la misma posición que el otro pero en diferentes caras de la placa.

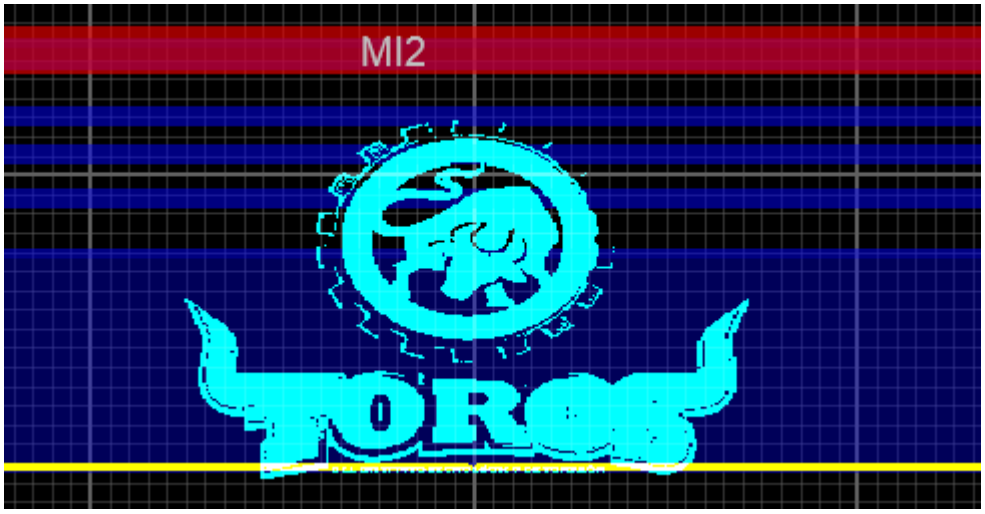


Imagen 187. Logotipo superior.

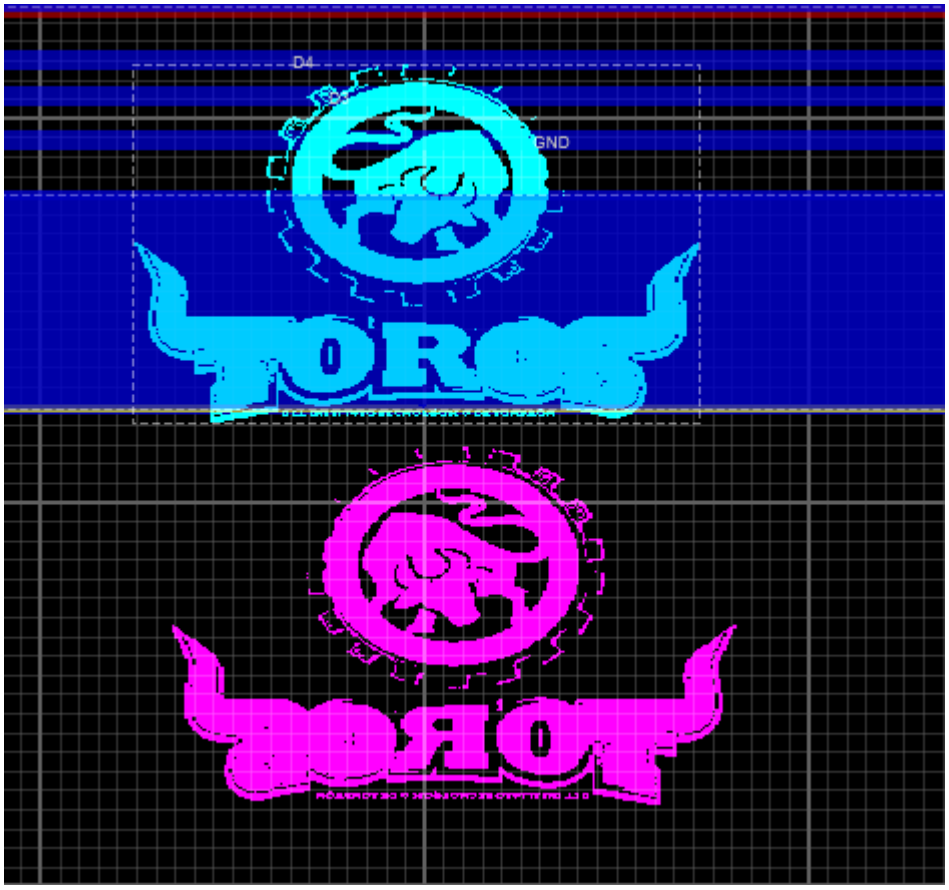


Imagen 188. Logotipo inferior.

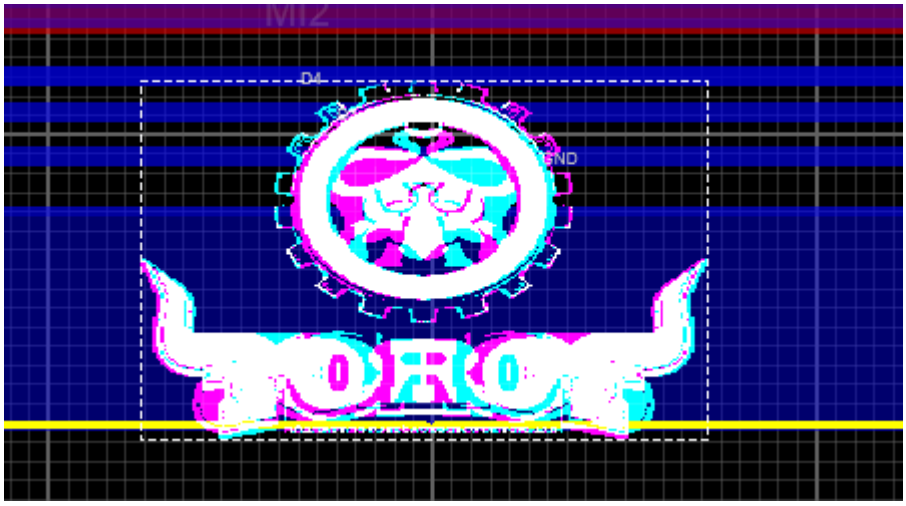


Imagen 189. Logotipo inferior y superior en la misma posición.

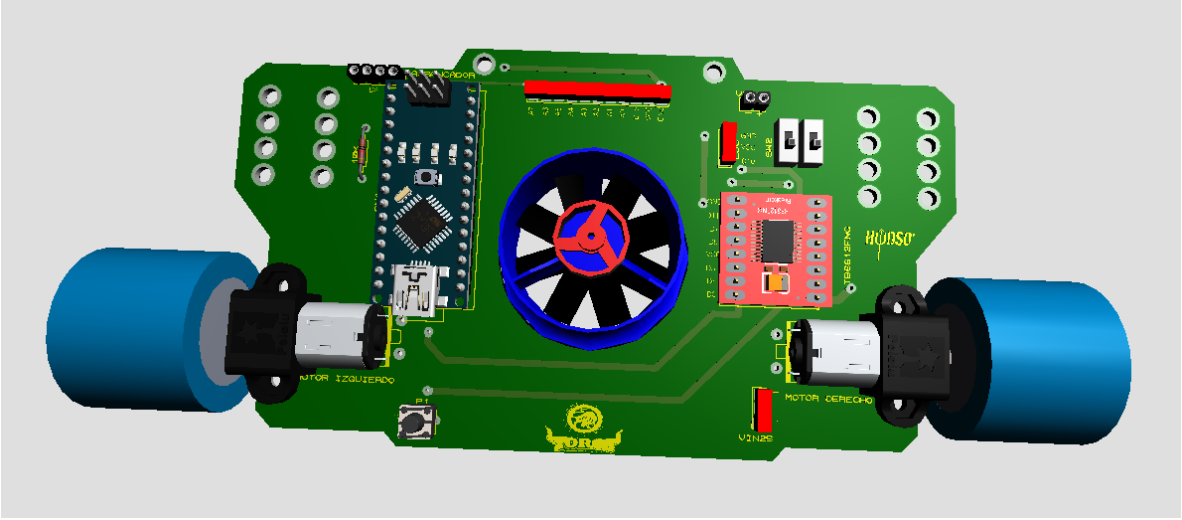


Imagen 190. Logotipo superior en placa 3D.

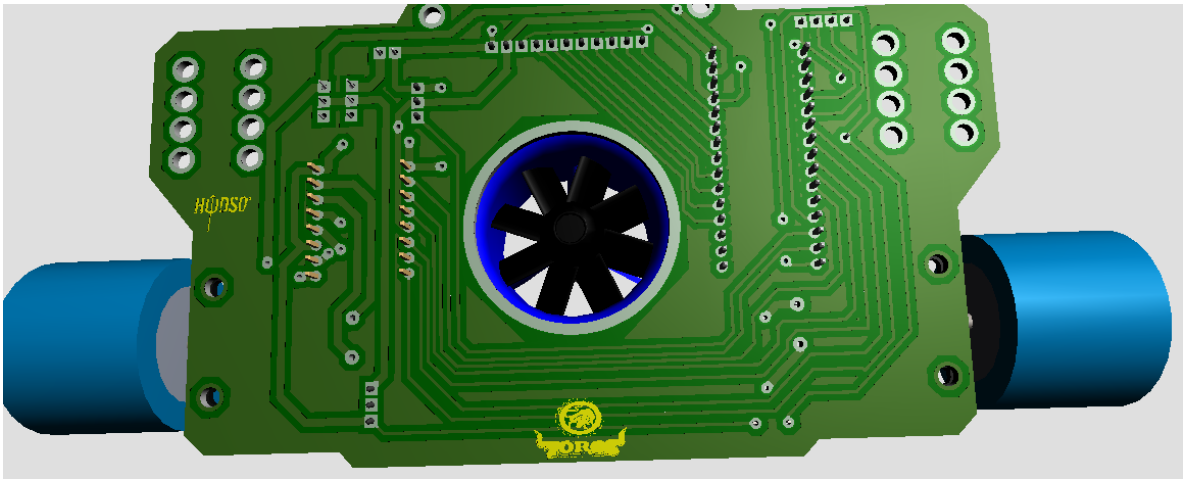


Imagen 191. Logotipo inferior en placa 3D.

Con los logotipos terminados se puede hacer un cambio de color para la placa, para el caso del Exia se consideró una placa con colores morados, siendo este el color principal de la placa total, color que aplica también para las vías de conexión interna de la placa.

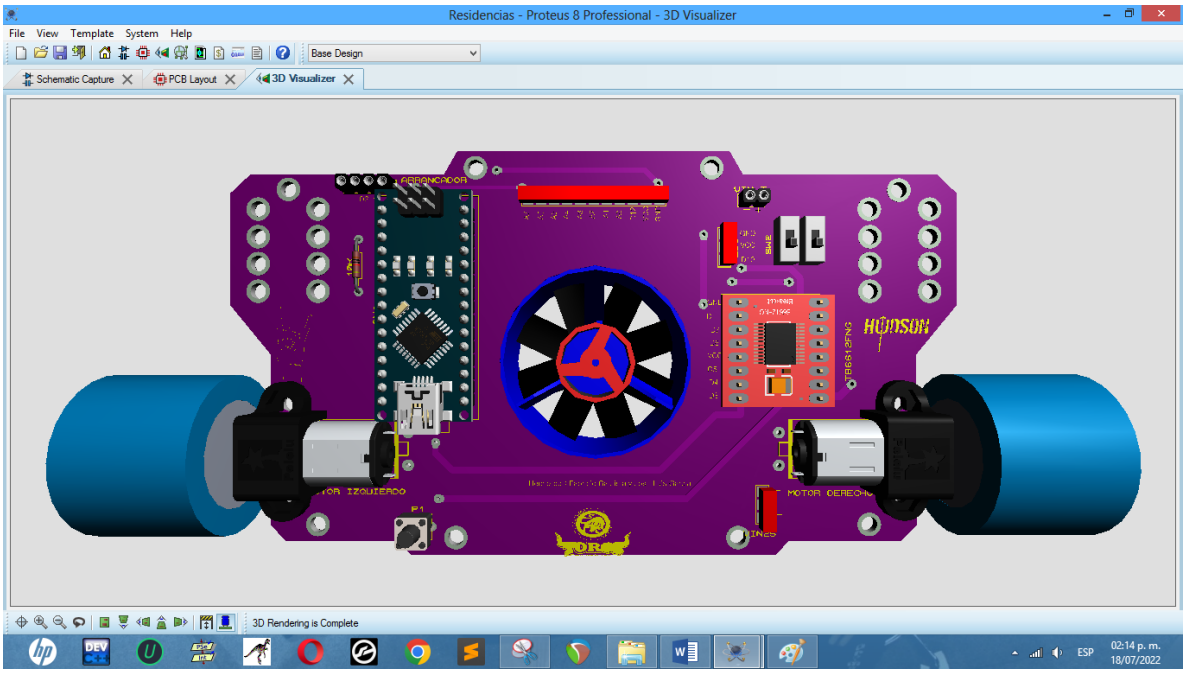


Imagen 192. Placa Exia superior.

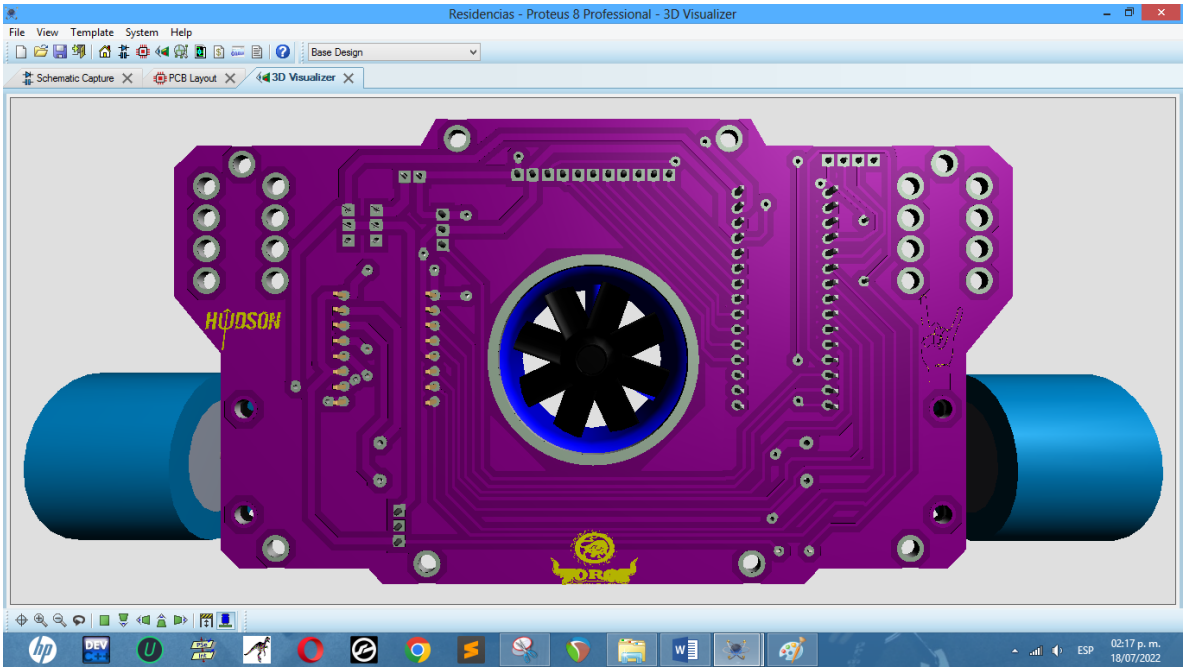


Imagen 193. Placa Exia inferior.

(Nota: Los huecos inferiores son para colocar una barra que evite una flexión en la parte inferior de la placa, que puede ser provocada por la turbina, pero no son necesarios para una placa de 1 mm de grosor y no fueron necesarios para el proyecto).

4.5.7 Documento gerber para la impresión de la placa

Los archivos Gerber son importantes en esta parte del proyecto porque son empleados por cualquier empresa dedicada a la impresión de placas (PCB), estos archivos serán realizados en base al diseño o los diseños realizados. Para obtener estos archivos lo primero por hacer es seleccionar la opción “Output” en la parte superior de la pantalla y en la ventana de opciones desplegada se deberá seleccionar la opción “General Gerber/Excellon Files”

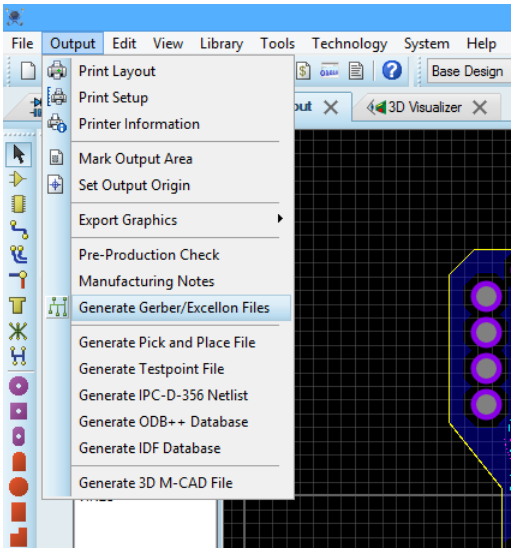


Imagen 194. Generando archivos Gerber.

Después de seleccionar esta opción Proteus arrojara una ventana que realizara un análisis de la placa para verificar que no contenga errores. En la búsqueda de errores el programa se encarga de hacer una inspección minuciosa de la placa, e informa de los errores encontrados, como puede ser la conexión o no conexión entre 4 pines del Arduino Nano, que como tal no es un error porque esas conexiones existen de forma interna dentro del Arduino Nano, pero el programa lo puede interpretar de esa manera e informar de este o cualquier otro error omitido.

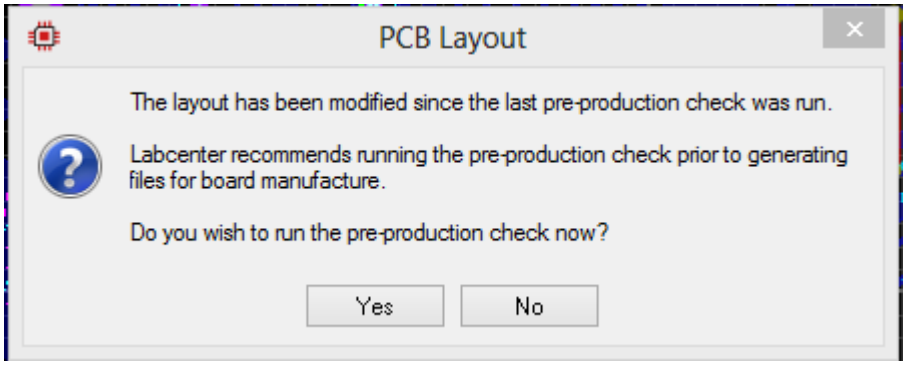


Imagen 195. Inspección de la placa.

Dando click en la opción Yes, el análisis se dará por iniciado y se desplegará una nueva ventana que comenzará a recopilar información e indicar los errores de la placa. Efectivamente en la revisión de la placa se puede observar con letras rojas un error el programa, que hace mención de las 2 conexiones no realizadas en el Arduino Nano, que para este caso se omite.

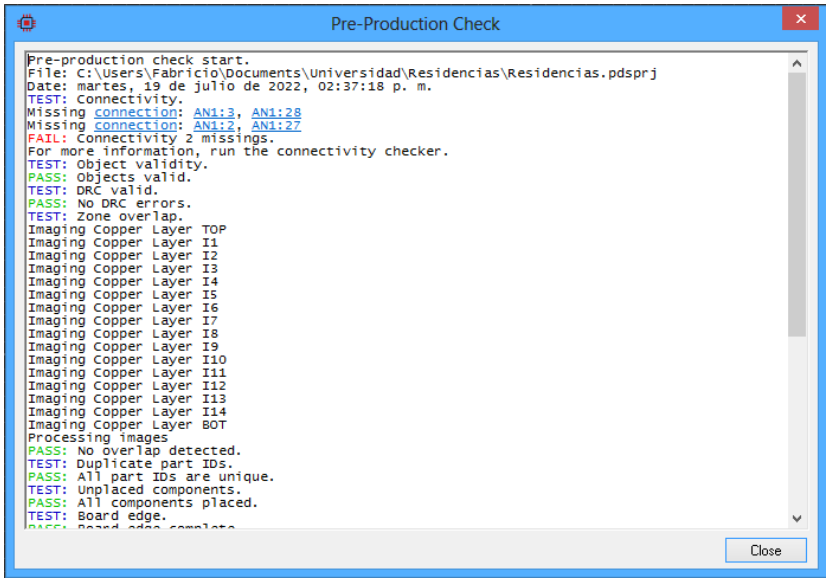


Imagen 196. Resultados del análisis para la placa 1.

Para la segunda imagen de resultados de la placa se puede observar que no existe ningún peligro alguno para la PCB y para su realización física, teniendo en total 16 pruebas realizadas por el programa, que se muestran en la parte baja del análisis marcadas en color azul como TEST y aprobadas o exitosas en color verde como PASS, teniendo como resultado un total de 16 pruebas realizadas, 0 errores y 1 falla.

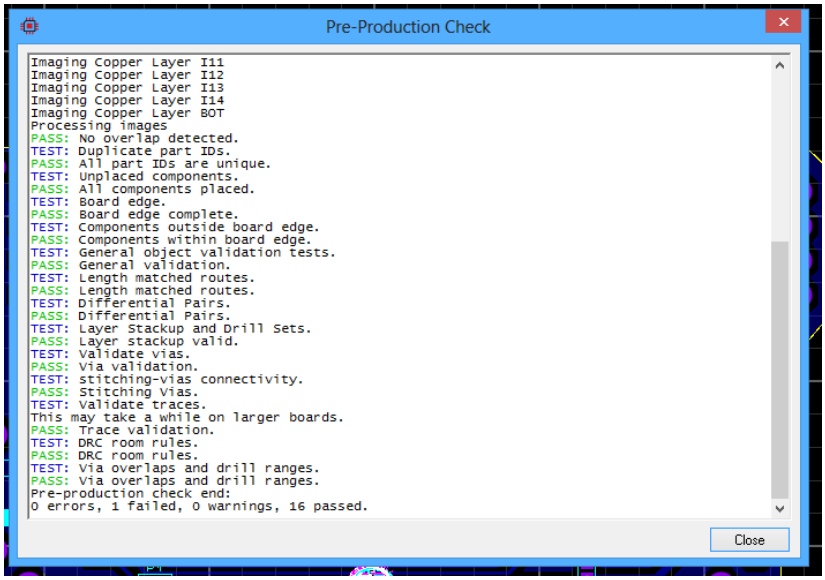


Imagen 197. Resultados del análisis para la placa 1.

Una vez obtenidos los resultados se puede dar “Close” a la ventana de pruebas para continuar con el proceso, nuevamente es lanzada una ventana de aviso que expone que si a pesar de la falla (las dos conexiones faltantes) se quiere continuar con el proceso para generar los archivos gerber, se debe dar click en YES para continuar.

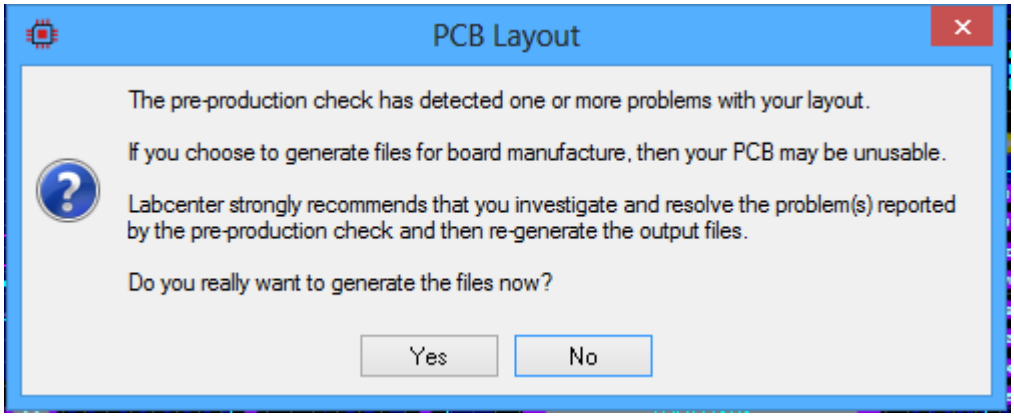


Imagen 198. Confirmación para archivos Gerber.

Ahora que se saben todos los problemas que tiene la placa una nueva ventana de Proteus saldrá a pantalla para solicitar una ubicación de guardado para el archivo Gerber, cada quien de manera

personal sabrá donde guardar estos archivos, pero cabe mencionar que es recomendable tener una carpeta especial para todos los documentos generados para el seguidor de línea, el cual resulte de fácil acceso para la localización de los mismo, en este caso esa configuración ya fue efectuada para el proyecto.

Generalmente los archivos Gerber que solicitan las empresas dedicadas a la elaboración de PCB's solicitan archivos comprimidos, por tanto se marcara la opción "Output to a single ZIP file". Posteriormente para el apartado Layers/ArtWorks que son las opciones que están ubicadas por debajo de estas palabras deberán dejarse seleccionadas casi todas, a excepción de 2 Top Paste y Bottom Paste. Para la siguiente fila se dejarán libres de Inner 1 hasta Inner 7 y marcadas únicamente Mech 1 y Drill (Mech 2 se queda libre). La tercera fila se queda libre en su totalidad. Para la rotación se debe dejar seleccionada la opción X Horizontal, para Reflection se debe dejar seleccionada la opción Normal. Para las unidades del archivo (File Units) lo ideal sería tenerlo en metric (mm) o sea en métrico, esto con la finalidad de realizar un análisis de la placa de forma final en los analizadores que tienen paginas especializadas en la realización de placas PCB. Para el formato Gerber (Gerber Format) se debe dejar seleccionada la opción Gerber x2. En la opción Slotting se deja todo tal cual, y para la opción "Run Gerber Wiewer When Done" debemos seleccionar su casilla. Por último en la parte inferior derecha se debe seleccionar una resolución de 500 dpi para una buena resolución, también se puede dejar una resolución mayor, por ejemplo 1000 dpi, que ofrece una mejor calidad de imagen y la única diferencia es el tamaño o peso del archivo. El nombre de la placa se puede designar al principio, y eso depende del nombre de cada archivo. En la imagen siguiente se puede observar toda la información que se ha mencionado.

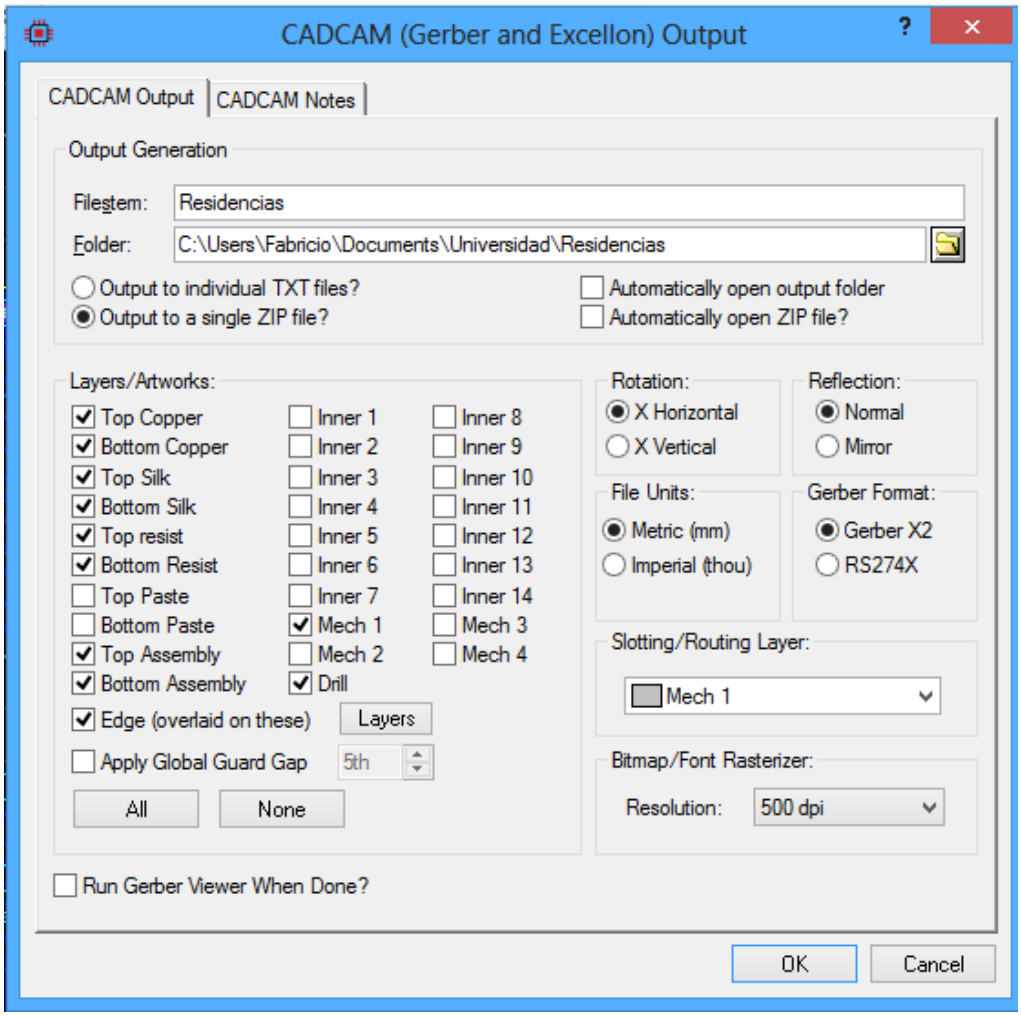


Imagen 199. Características del archivo Gerber.

Posteriormente a dar OK a esta ventana, el programa procesara toda la solicitud de la información colocada, e inmediatamente será arrojada una nueva ventana (Gerber View) con todas las capas realizadas para la placa y simplemente se tiene que dar OK.

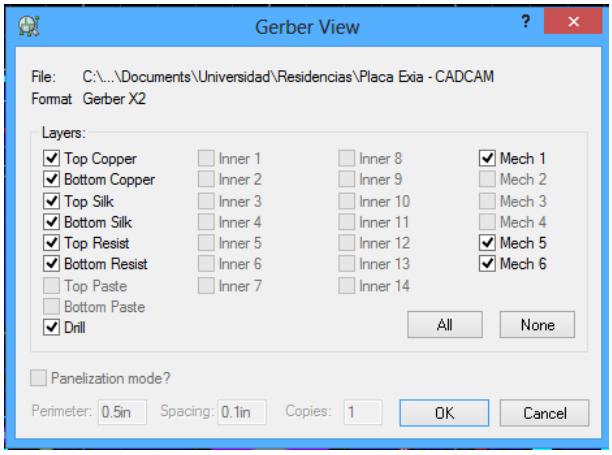


Imagen 200. Capas de la placa para archivo Gerber.

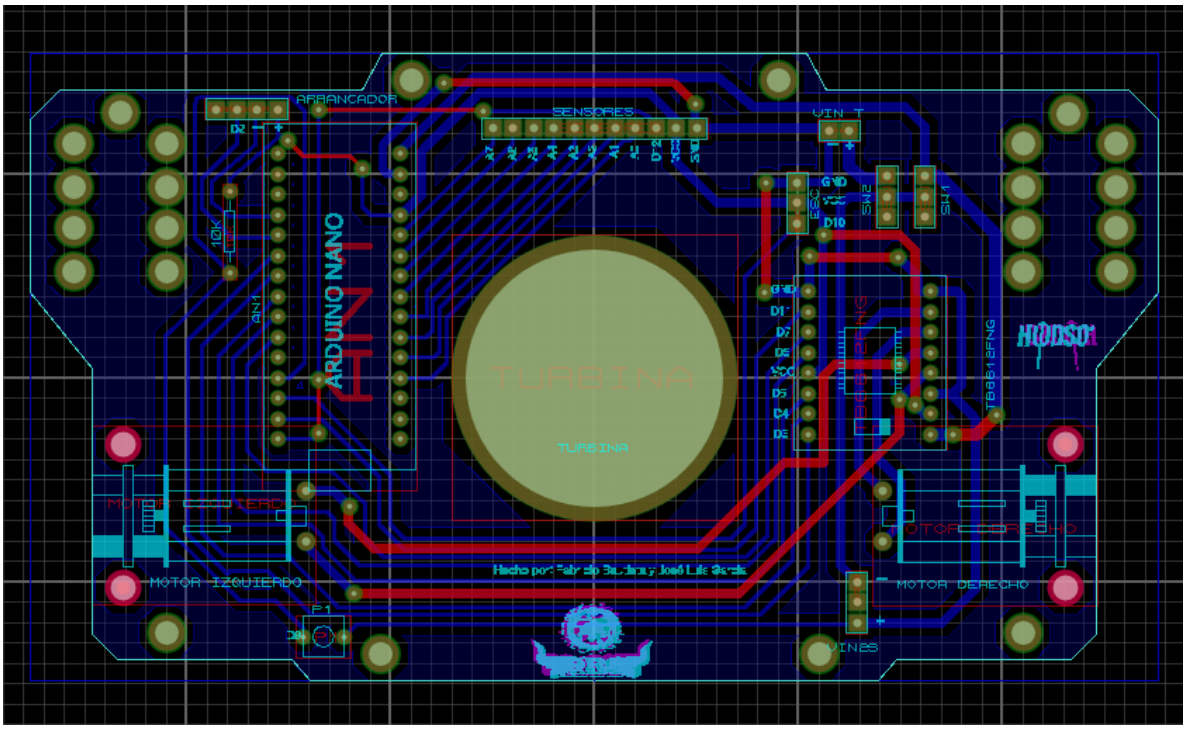


Imagen 201. Placa para archivo Gerber.

La placa mostrada es la misma placa desarrollada en este documento, pero con el proceso de archivo Gerber finalizado, cuando este proceso ha terminado el programa en inmediato lanza una nueva ventana totalmente ajena a la ventana de trabajo del PCB Layout donde se observa la placa

junto con todas sus capas, todos los componentes introducidos en ella, etc. Esta nueva ventana se llama Gerber Wiewer.

4.6 Cómo solicitar placas por internet

Para solicitar placas se puede contactar de forma personal con una empresa dedicada a este trabajo si se tiene conocimiento de alguna, para el caso del proyecto así fue, se contactó por medio de un ingeniero con una empresa dedicada a la elaboración de placas PCB y los requerimientos que fueron solicitados fueron: el archivo Gerber, el grosor de la placa que fue de 1 mm pero también se tenía la opción de 0.6 mm y el color de la placa, además del pago correspondiente por la placa, como tal las empresas en la mayoría de los casos fabrican las placas a partir de un pedido de 5 piezas, lo cual es bueno en caso de requerir más de una placa pero a la vez negativo en costos sobre el precio final de la placa del seguidor, resultando en no pagar solo una, sino de varias. En caso del proyecto se pagaron $1050 por 5 PCB. Sin embargo también se puede recurrir a solicitar placas por medio de internet, a continuación se presenta la forma de solicitar placas con la empresa de origen chino JLCPCB. Para solicitar las placas lo primero es acceder a su página web, entonces por medio de un navegador se debe colocar el nombre de la empresa en la barra de búsqueda “JLCPCB”, se mostrarán varias opciones, pero la correcta será la que responde al nombre de “JLCPCB PCB Prototype & PCB Fabrication Manufacturer”. La opción a elegir es la primera que se muestra en la imagen siguiente:

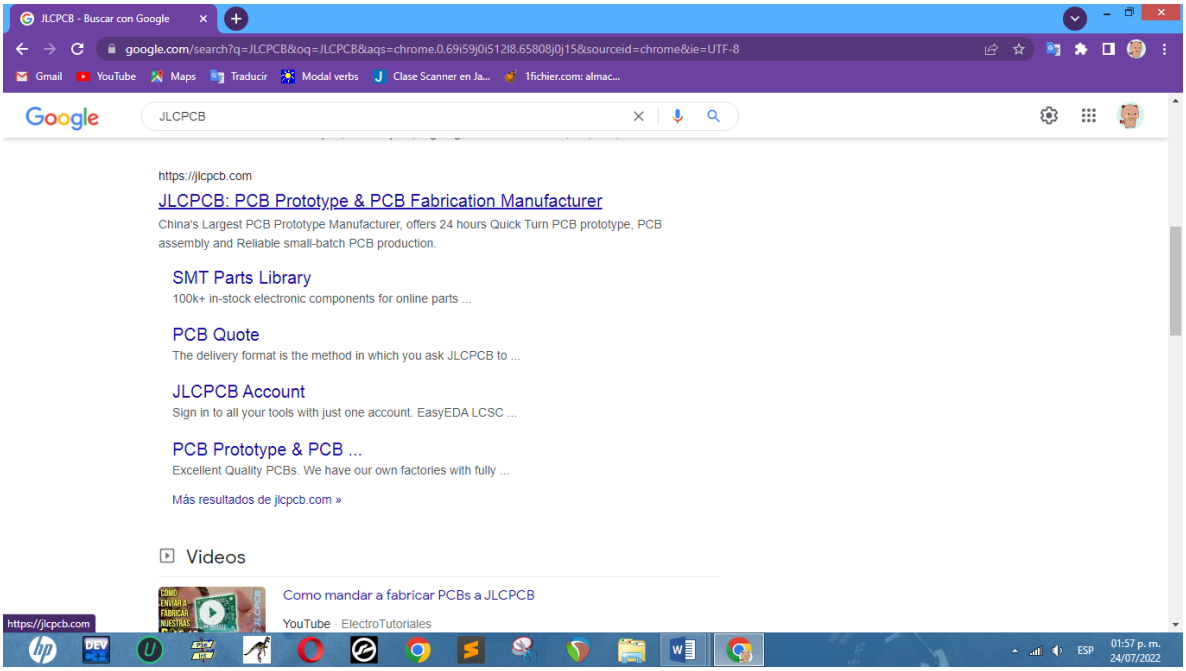


Imagen 202. Búsqueda de la página de JCVPCB.

Una vez dentro de la página se puede observar una pantalla principal que muestra bastantes opciones como: qué empresa es, capacidades, apoyo, recursos, costos, inicio de sesión, etc. En esta última es recomendable contar con una cuenta o usuario para la página que otorga el beneficio de realizar cotizaciones de placas, pero no solo eso sino también la opción “Recursos” que permite realizar el diseño de placas en “EasyEDA” que es una herramienta que ofrece la empresa para realizar placas de manera gratuita, entre otras opciones. En caso de no contar con una cuenta, la pantalla principal brinda la opción de una cotización instantánea (opción a elegir), en la parte izquierda se solicitara el archivo Gerber para la placa, se elige esa opción para abrir una ventana de búsqueda para seleccionar y abrir el archivo, se debe buscar el archivo en la ubicación donde fue guardado y con el nombre designado, al momento de abrir el archivo, la página cargara el archivo Gerber y de inmediato se podrá visualizar tanto la cara frontal como trasera, y se mostrarán otras especificaciones de la placa. Entre las cuales se encuentran: el material del que está hecha la

placa, número de capas (en este caso 2), dimensiones, número de placas que van desde 5 hasta 80,000 que en base a esto define un precio de producción. Características a elegir: tipo de producto o su utilización, formato simple, espesor de la placa que para el caso de los seguidores de línea velocistas es muy importante contar con un espesor de entre 0.6 hasta 1.2 mm. En el caso del proyecto se eligió un espesor de 1 mm tanto por cuestiones de peso como de resistencia para evitar dobleces debido a la fuerza ejercida por la turbina. Color de la PCB, siendo morado el color elegido para la placa se puede tener una la placa de cualquier color, que no afecta en nada al funcionamiento y es más una cuestión estética, acabado de la superficie HASL con plomo, peso del cobre 1 oz, dedos de oro la opción sería no, prueba de sonda, esta prueba no afecta en el precio así que lo mejor es dejar activada esta opción y lo demás se queda tal cual.

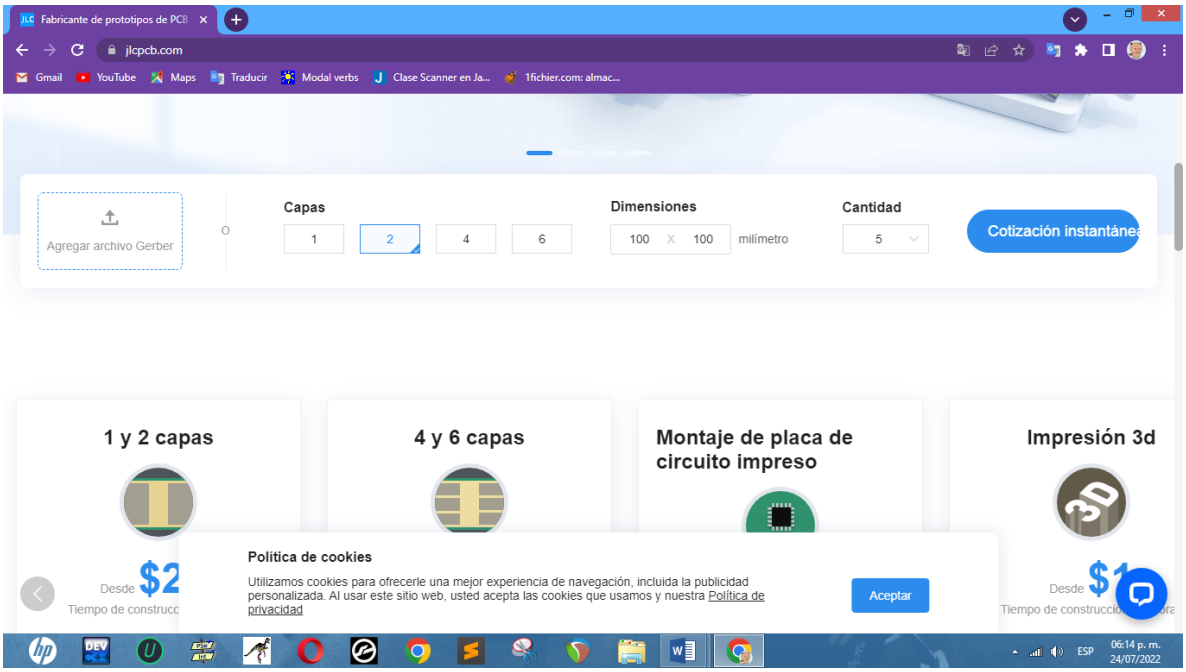


Imagen 203. Página principal de JCVPCB.

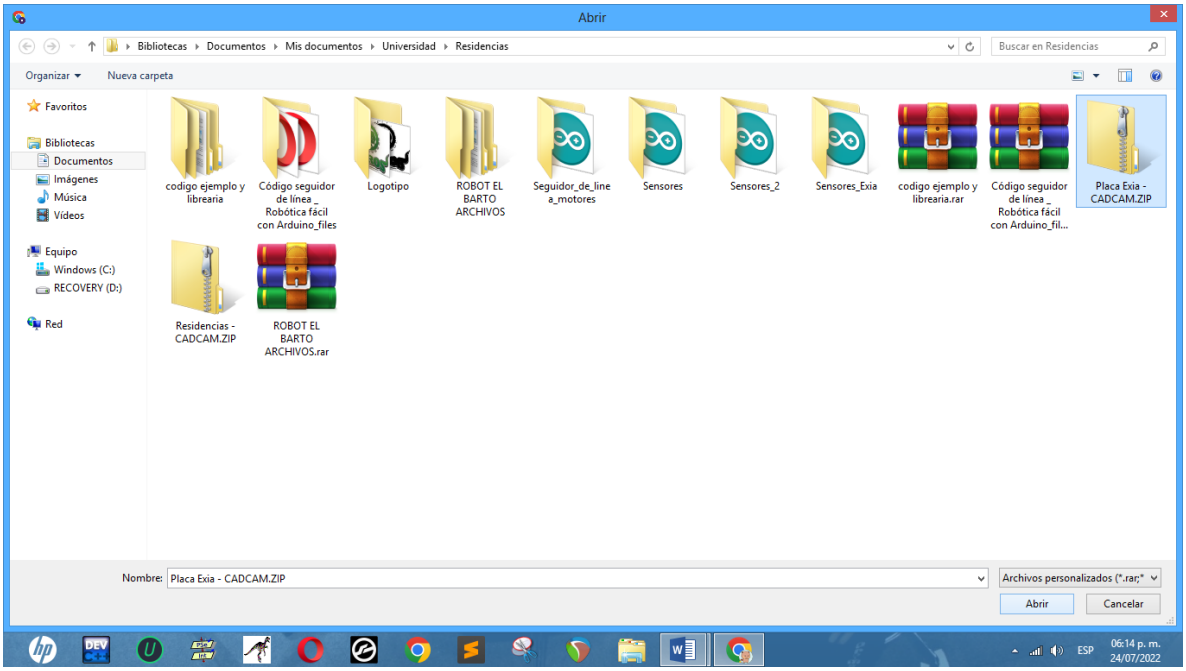


Imagen 204. Búsqueda de archivo Gerber de JCVPCB.

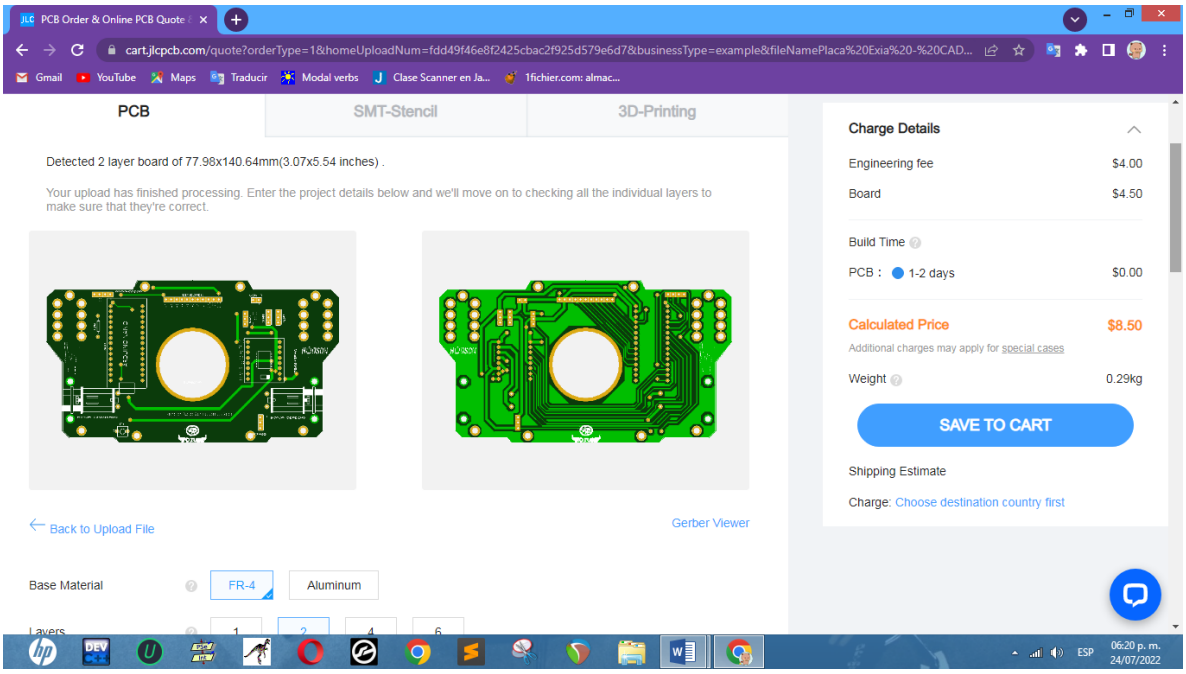


Imagen 205. Visualización de la placa en ambas caras.

La página también permite la opción de una visualización de la placa de forma más detallada, opción que se encuentra debajo a la derecha de las imágenes de ambas caras de la placa que

responde al nombre de “Gerber Wiewer” donde se tiene un visualizador con todos los componentes que colocados a la placa, que desafortunadamente requiere tener una cuenta para acceder a esta opción o de lo contrario el acceso es denegado.

También a lado derecho se podrá observar el costo de la placa, que para este caso es aproximadamente de $8.50 dólares, que convertido a pesos son aproximadamente $175 pesos mexicanos, por un total de 5 placas PCB, pero este costo es sin mencionar el envió, y el envió dependiendo del lugar puede ser mayor o menor, que para México no se tiene una cantidad exacta puesto que no se eligió la opción de JCVPCB para la placa, pero una vez registrado se puede agregar el lugar de destino y se tendrá una cantidad específica del producto con el envió, cabe mencionar que tampoco se mencionan cargos aduanales adicionales. Para realizar el pago se solicita una tarjeta de débito o crédito y es decisión de cada quien el método de pago para conseguir la PCB, posteriormente será solicitada información correspondiente a dirección, código postal y demás datos personales. Prácticamente este sería el proceso para solicitar PCBs por internet.

4.7 Ensamble del seguidor

Para el ensamble del seguidor se deben tener todos los componentes mencionados en la tabla de presupuestos, además de la PCB ya sea artesanal o de empresa, juntos a estos componentes será necesario cautín, estaño, pasta y en caso de contar con ella una tercera mano para soldar.

Lo primero por realizar es soldar espadines hembra para el Arduino Nano, el puente H, para el modulo arrancador, y para el ESC, para el caso de los motores y la fuente de alimentación de la turbina deben ser espadines macho, para este momento también se deben soldar los dos Switch, el resistor, la conexión para la batería, pulsador y los motores cuidando de estos últimos su polaridad.

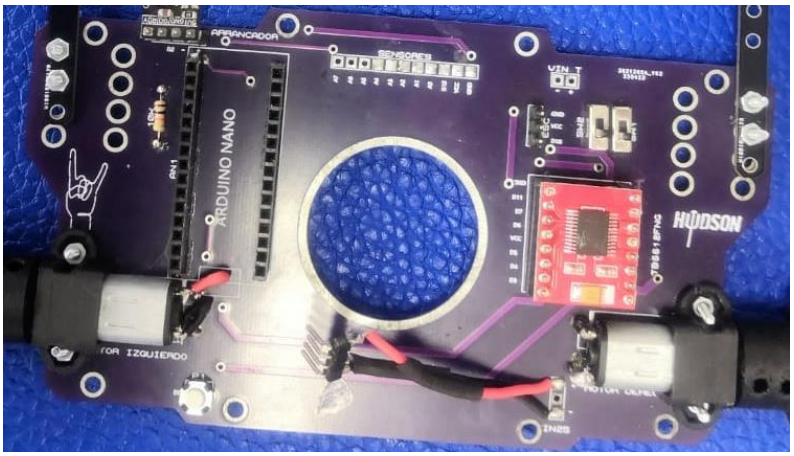


Imagen 206. Vista superior del seguidor con los espadines colocados..

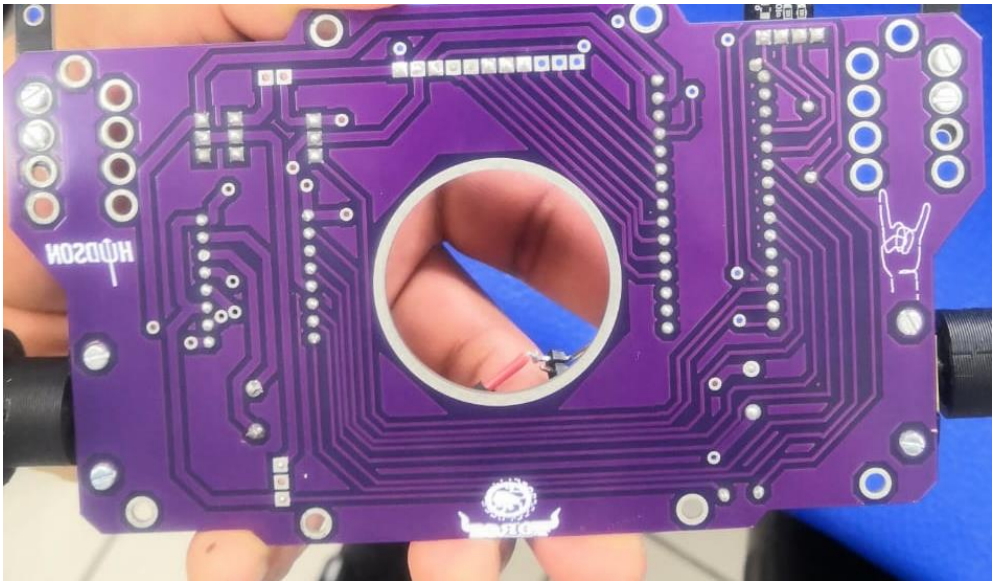


Imagen 207. Vista inferior del seguidor con la soldadura de los espadines.

Después se debe instalar la turbina, para el caso de la turbina por cuestiones de potencia y a criterio de cada quien se debe utilizar un pegamento con potencia que la sujete a la placa de modo que al ser activada la turbina no salga volando del seguidor, cosa que sería contraproducente, en el caso de la turbina del proyecto se utilizó pegamento 5000 pero con una base de plástico del diámetro de la turbina para realizar mayor sujeción, y es a consideración de cada quien lo que utilice para tenerla fija. Al momento de conectar la turbina se tiene que conectar el ESC a ella y para esto se recomienda revisar las polaridades del ESC y de la turbina para conectar correctamente los cables, después se tiene que hacer la conexión de estos cables a su respectivo espadín que está marcado en la turbina. En el caso de la corriente el cable rojo va a positivo y el negro a negativo.

Con las conexiones realizadas de la turbina se puede proceder a realizar las conexiones del puente H, el Arduino Nano y la batería cuidando que los dispositivos estén en sintonía con las imágenes de la placa de los mismos dispositivos para evitar que al momento de emitir corriente por la placa alguno de ellos se queme.

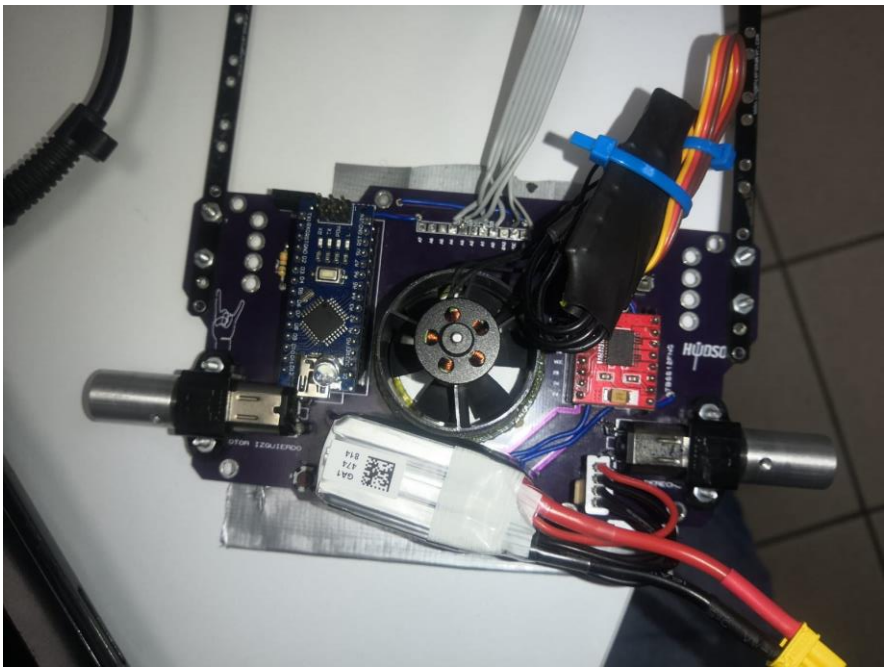


Imagen 208. Seguidor con la turbina, puente H, Arduino Nnano, Motores.

Lo siguiente es realizar la conexión de la barra de sensores, que en caso de utilizar una barra de Ingeniero Maker las conexiones son prácticamente directas, revisando las inscripciones que tiene la barra deben coincidir con las de la placa, en el caso de esta barra se inicia por el extremo derecho iniciando por GND para GND de la placa y se termina en OM que para la placa seria realiza la última conexión en A4, siendo todas las conexiones directas con su homónimo.

Para el tema de una barra de sensores diferente se tendrían que revisar las conexiones de la barra y unirlas a su homónimo en la placa, para la siguiente imagen se cambió la barra de Ingeniero Maker por una tipo Phoniex y las conexiones con la placa fueron las siguientes iniciando de derecha a izquierda: S2 a A2, S3 a A3, VCC a VCC, OP a A4, GND a GND, S1 a A1, S0 a A0.

Una vez terminadas las conexiones lo que queda por hacer es conectar el las barras que sujetan a la barra de sensores. Colocando los tornillos y sus tuercas en las perforaciones de la placa que se

consideren mejor, por este motivo se realizaron varias perforaciones en la placa, y también considerando que el seguidor no debe pasar de 25 cm de largo.

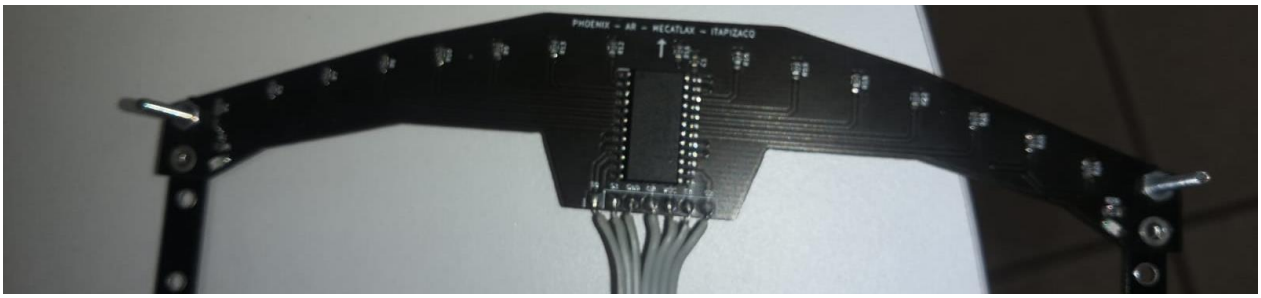


Imagen 209. Barra de sensores tipo Phoenix diseñada por el Ing. Eduardo Castillo.

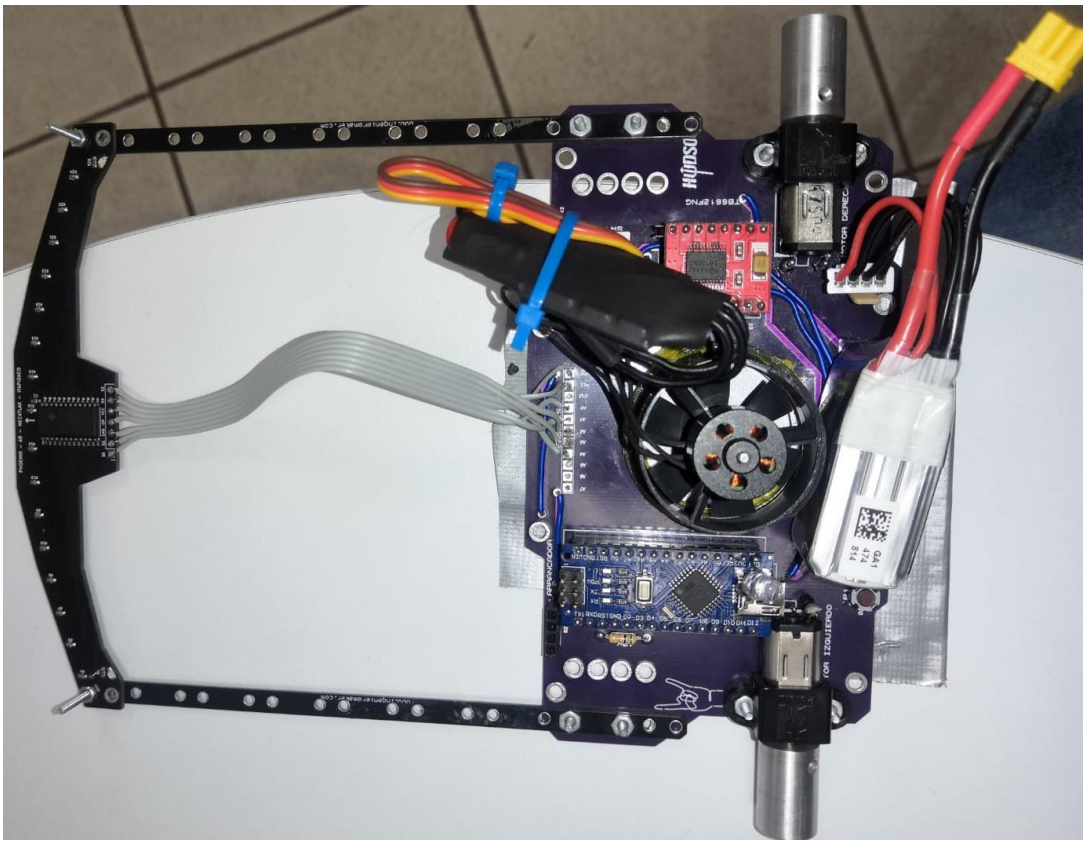


Imagen 210. Seguidor de línea con barra de sensores.

Al seguidor se le colocaron trozos de cable en las perforaciones que son las conexiones superiores de la placa haciendo alusión a una placa de una cara o artesanal, pero son innecesarias para una placa de dos caras, aunque se tengan estas perforaciones para las conexiones superiores. Como último solo se deben colocar las llantas de caucho sobre los motores y el seguidor estará listo para correr, claro ya con el programa dentro del Arduino Nano, tema a estudiar en el siguiente capítulo.

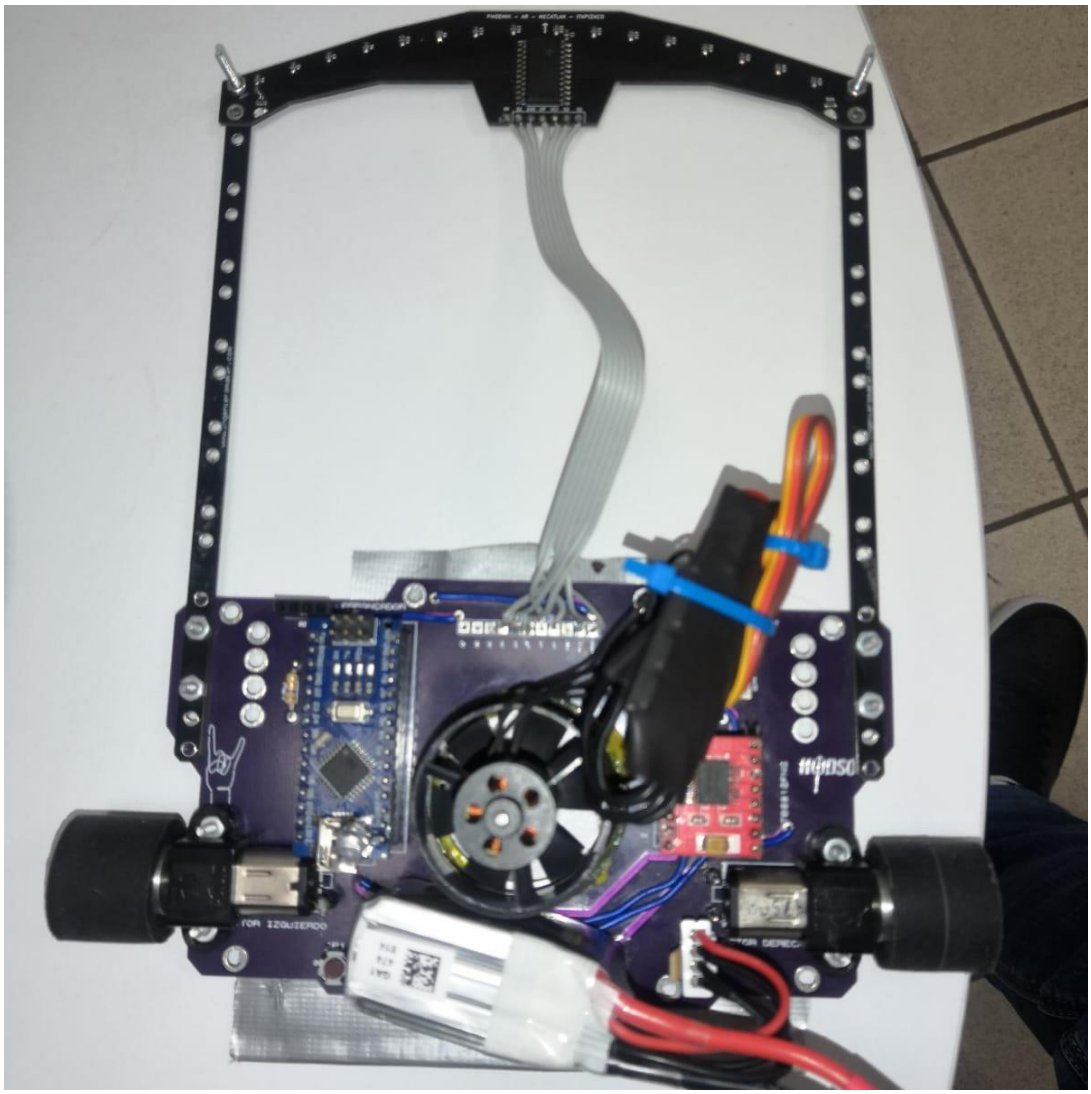


Imagen 211. Seguidor de línea armado por completo.

CAPÍTULO 5. PROGRAMACION E IMPLEMENTACION DE UN PIDOF PARA EL SEGUIDOR DE LINEA VELOCISTA

Para realizar una programación de forma correcta se deben poseer conocimientos previos de esta disciplina, el programa a utilizar es "Arduino" y es utilizado para la programación de estos microcontroladores, en este caso el Arduino Nano utilizado en el proyecto, y como en cualquier lenguaje de programación, las bases son las mismas, salvo algunas pequeñas cosas que cambian dependiendo del programa y lenguaje a utilizar, para programar en Arduino se puede tener como referencia la programación en "C++". Para el desarrollo de la programación del proyecto se ha optado por realizar programas enfocados a partes especificas del seguidor; las partes más importantes como lo son, los motores y la regleta de sensores, para finalizar con un programa que englobe estas dos partes, más el PID. Por tanto se inicia con un programa para los motores con la finalidad de observar el comportamiento entre ambos motores y su funcionamiento de forma pareja. Para posteriormente continuar con la regleta de sensores, y de igual forma observar el comportamiento de cada uno de los sensores.

5.1 Programación de motores

Para realizar la programación de los motores se toma en cuenta tanto a los motores como al Driver, en este caso el puente H TB6612FNG. Justamente se debe partir por el puente H, y recordar cual es la distribución de los pines para los motores. Basados en el orden de pines del puente H se puede observar que el pin AO1 y AO2 del puente H están conectados al MI2 y MI1 del motor izquierdo. Pero los pines encargados del control de esta salida son los pines AIN2 y AIN1. El PWMA será el encargado de regular la velocidad, que quiere decir esto, pues que los pines D5 (Adelante) y D4 (Atrás) serán los encargados de la dirección del motor o sea, hacia adelante y hacia

atrás, y el pin D3 (PWMA) será el encargado de regular la velocidad del motor, por ejemplo 100 RPM ya sea hacia adelante o hacia atrás, eso para el motor izquierdo. Para el motor derecho el pin BO2 y BO1 del puente H están conectador al MD1 y MD2 del motor derecho. Y los pines encargados del control de esta salida son los pines BIN1 y BIN2. El PWMB será el encargado de regular la velocidad, siendo el pin D6 (Adelante) y D7 (Atrás) los encargados de la dirección del motor, y el pin D11 (PWMB) encargado de regular la velocidad del motor derecho.

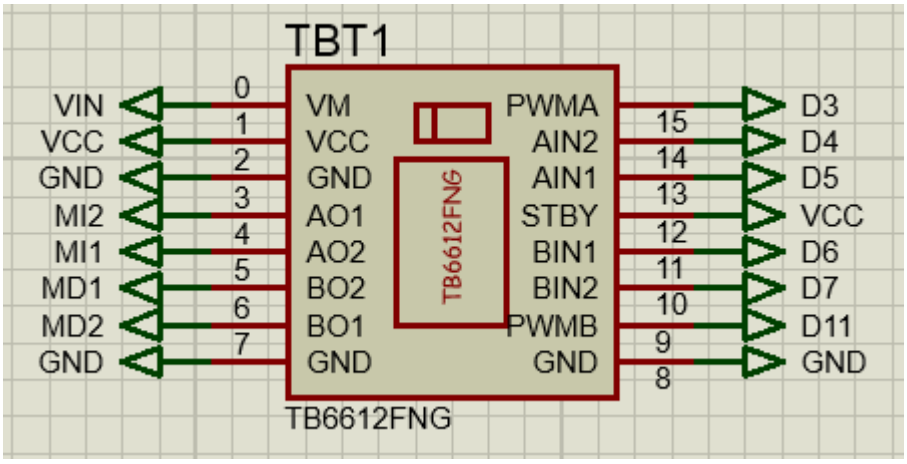


Imagen 212. Visualización de la placa en ambas caras.

Para iniciar con la programación se abre el programa “Arduino” como archivo nuevo y se comienza con la programación. Para iniciar se deben definir las variables de los motores, en este caso las variables de los pines.

Se inicia con el motor izquierdo, para el motor izquierdo se utiliza el siguiente código:

```
#define  pwmi  3 //PWMA MOTOR IZQUIERDO

#define  izq1  5

#define  izq2  4

#define  pwmd  11 //PWMB MOTOR DERECHO
```

#define der1 6

#define der2 7

En principio se define la variable para el PWM del motor izquierdo, junto a su pin correspondiente (pin 3) posterior a ello las salidas de este motor, izq1 que será el movimiento hacia adelante con el pin 5 y izq2 que será el movimiento hacia atrás con el pin 4. Continuando con el motor derecho lo mismo, se inicia por el PWM con el pin 11, y con las salidas der1 para el movimiento hacia adelante con el pin 6 y der2 para el movimiento hacia atrás con el pin 7. Delante de cada PWM (izquierdo y derecho) se ha dejado una nota que indica para que motor es cada PWM esa nota se utiliza bastante y se pone utilizando doble slash antes de la nota.

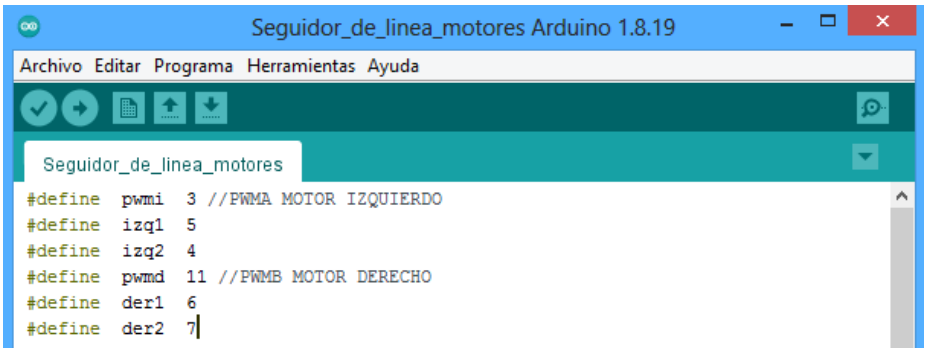


Imagen 213. Definiendo variables PWM y de salida.

En el void set up se van a declarar los pines de salida que van a funcionar de manera digital, en este caso las direcciones de ambos motores deben declararse como salida debido a que se activaran o desactivaran dependiendo de la dirección, y esto es fácil de analizar de acuerdo a la lectura que reporte la barra de sensores, por medio de la programación se le indicara a los motores en qué dirección girar y a qué velocidad para reestablecer la posición en el circuito o pista siguiendo la línea marcada, por tanto si el seguidor de línea sale por un lado de la pista este dato será enviado a través de la barra de sensores hacia el microcontrolador y este a su vez enviara una acción en

respuesta a esta situación para efectuar los ajustes de dirección y velocidad necesarios en los motores para volver a la trayectoria principal. En cambio los pines para los PWM no son declarados como salida porque funcionan como salida analógica.

Los pines se declaran de la siguiente manera:

```
pinMode (izq1, OUTPUT);

pinMode (izq2, OUTPUT);

pinMode (der1, OUTPUT);

pinMode (der2, OUTPUT);
```

Para la declaración de pines se utiliza la variable pinMode seguido de la dirección de cada uno de los motores, seguido de OUTPUT para la declaración de salida, este proceso se repite para las 4 direcciones que presenten los motores, dos por cada uno.

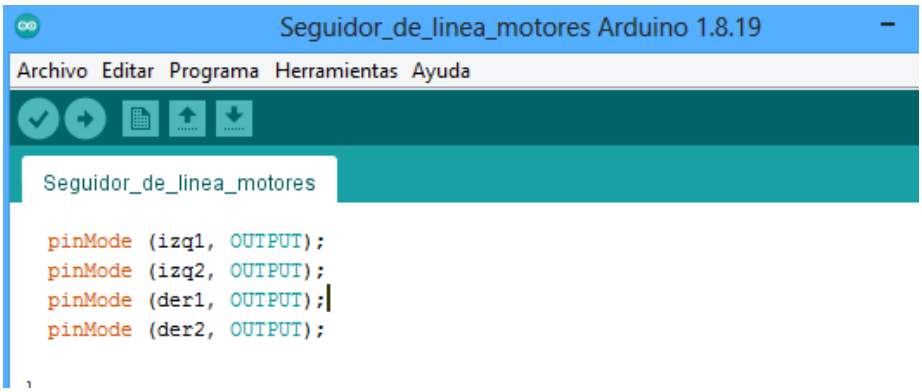


Imagen 214. Declaración de salidas.

Una vez declarados los pines de salida se realiza una nueva función la cual se denominó como “Motores” a la que se le introducen valores positivos para que los motores giren hacia adelante, en este caso el seguidor avanza, y valores negativos para que lo motores giren hacia atrás, por ende el

seguidor retrocede. Para crear esta función se utiliza void loop iniciando con un nuevo void seguido de la palabra motores para crear la función, dentro de la función se agregan dos valores que serán izq para el motor izquierdo y der para el motor derecho. Delante de la función se puede colocar una nota que indique cual es la velocidad mínima y máxima a la cual operan los motores. En este caso es de 0 a 255 o de 0 hasta -255 para velocidades de retroceso. La función es la siguiente:

void motores (int izq, int der){ //VELOCIDAD REGULADA DESDE 0 HASTA 255 o 0 hasta -255

```
}
void motores (int izq, int der){ //VELOCIDAD REGULADA DESDE 0 HASTA 255 o 0 hasta -255
```

Imagen 215. Declaración de variable “motores”.

Ahora que se ha designado una nueva función para los motores, y se sabe cómo funciona y como son las conexiones en el driver (Puente H), se pueden introducir los valores para los motores y definir si son positivos o negativos, resultando en un giro hacia adelante o un giro hacia atrás. Para ello debajo de la función de motores se debe escribir una nota que indique que es para el motor izquierdo y debajo de la nota se debe colocar una variable if, dentro de ella se coloca izq y un menor igual a 0. Para esta primera función si el valor es mayor a 0 el motor izquierdo tendrá que ir hacia adelante habilitando al pin 5 definido al inicio desactivando el pin 4 que es un giro hacia atrás. Debajo del if se termina de colocar la condición con la variable digitalWrite teniendo primero izq1, HIGH (Encendido) y después izq2, LOW (Apagado) dando como resultado que a valores positivos el motor debe girar hacia adelante para avanzar. Pero aún falta definir el caso contrario para que el motor pueda retroceder, para esto se utiliza la variable else seguida de la variable digitalWrite teniendo nuevamente de inicio a izq1 pero ahora con LOW (Apagado) seguido de

izq2, HIGH (Encendido) para que cuando se tenga un valor negativo el motor gire en sentido contrario y en vez de avanzar, retroceda.

Para el valor positivo no existe complicación alguna porque el PWM admite valores positivos, pero para los valores negativos no funciona igual, dentro de esta programación se sabe que el valor es negativo, pero el PWM no distingue valores negativos en la velocidad, por lo que se agrega al código izq=izq*(-1) para volver el valor positivo. Entonces al multiplicar el valor del PWM en la función else se asegura que el valor sea positivo y la dirección cambie, prácticamente se juega con la lógica del programa y del sistema, porque se recibe un valor negativo el cual no es admitido por las salidas analógicas, pero al multiplicarlo por el -1 se convierte en positivo y es admitido por el PWM, en tanto, se puede concluir que se implementan valores positivos y negativos pero que los valores negativos se convierten a positivos para tener una rango de velocidad, porque en principio el programa ya ha admitido el valor negativo y ha cambiado la dirección de giro del motor, después se ha cambiado el signo para tener un valor de velocidad que puede ser asimilado por el PWM. Por último se debe asignar una variable para la velocidad del PWM, para esto se utiliza la variable analogWrite seguido de pwmi, izq para la velocidad designada a los motores. Este mismo proceso se repite para el motor derecho, pero con der1 y der2. El código es el siguiente:

```
////////////////MOTOR IZQUIERDO////////////
  if(izq>=0){
    digitalWrite(izq1, HIGH);
    digitalWrite(izq2, LOW);
  }
```

```
else{
  digitalWrite (izq1, LOW);
  digitalWrite (izq2, HIGH);
  izq=izq*(-1);
}
analogWrite(pwmi,izq);

////////////////MOTOR DERECHO///////////////

 if(der>=0){
  digitalWrite(der1, HIGH);
  digitalWrite(der2, LOW);
}
else{
  digitalWrite (der1, LOW);
  digitalWrite (der2, HIGH);
  der=der*(-1);
}
```

```
  analogWrite(pwmd,der);

}
```

```
void motores (int izq, int der){ //VELOCIDAD REGULADA DESDE 0 HASTA 255 o 0 hasta -255
  ////////////////MOTOR IZQUIERDO///////////
  if(izq>=0){
    digitalWrite(izq1, HIGH);
    digitalWrite(izq2, LOW);
  }
  else{
    digitalWrite (izq1, LOW);
    digitalWrite (izq2, HIGH);
    izq=izq*(-1);
  }
  analogWrite(pwmi,izq);

  ////////////////MOTOR DERECHO///////////////

   if(der>=0){
    digitalWrite(der1, HIGH);
    digitalWrite(der2, LOW);
  }
  else{
    digitalWrite (der1, LOW);
    digitalWrite (der2, HIGH);
    der=der*(-1);
  }
  analogWrite(pwmd,der);
}
```

Imagen 216. Sentido de giro de los motores.

Para verificar lo realizado, se utiliza la función void loop, debajo de la variable se debe colocar la palabra motores seguido del valor de velocidad para el giro de estos. Esta velocidad no es en conjunto, o sea no es una velocidad para ambos motores, sino que es una velocidad por cada motor. Como en la variable motores primero se escribió izq y después der, se entiende que la primera velocidad será para el motor izquierdo y la segunda para el derecho, separadas por una coma. Justo con esta función se puede definir la velocidad de giro de los motores y que esta velocidad sea positiva o negativa, adelante o atrás. El código es el siguiente:

```
void loop() {
```

motores(-10,-30); //(MOTORES IZQ,DER)

```
void loop() {
  motores(-10,-30);  //(MOTORES IZQ,DER)
```

Imagen 217. Velocidad de giro de los motores.

Esto no funciona así para el programa final, para el programa final se busca que con tan solo una velocidad ambos motores giren a la par, pero como ya se ha mencionado para este primer programa se busca hacer un análisis en el comportamiento de los motores de forma individual, observando su comportamiento tanto positivo como negativo, en uno o en ambos motores.

Ahora bien, con lo realizado hasta el momento queda casi por concluido el primer programa para los motores, esto que se ha realizado sirve para verificar que los pines que se seleccionaron para los giros de motor sean correctos ¿cómo es esto? bueno pues, lo primero es cambiar el valor de motor izquierdo por un valor de giro pequeño, bien puede ser de 20, para probar que gire hacia adelante, si es que el motor gira hacia adelante entonces la selección de pines fue correcta, después se puede cambiar por un valor negativo, por ejemplo: -20, para que el motor gire hacia atrás, si de igual forma gira de forma correcta entonces la selección de los pines de dirección ha sido acertada. Una vez probado el motor izquierdo se debe proceder a probar el motor derecho de la misma manera. Si por el contrario un motor gira hacia atrás con un valor positivo, entonces se deben intercambiar los pines de giro.

El programa completo para los motores es el siguiente:

```
#define  pwmi  3 //PWMA MOTOR IZQUIERDO

#define  izq1  5
```

```
#define  izq2  4

#define  pwmd  11 //PWMB MOTOR DERECHO

#define  der1  6

#define  der2  7




void setup() {


  pinMode (izq1, OUTPUT);

  pinMode (izq2, OUTPUT);

  pinMode (der1, OUTPUT);

  pinMode (der2, OUTPUT);



}

void loop() {

  motores(-10,-30);  //(MOTORES IZQ,DER)
```

```
    }

    void motores (int izq, int der){ //VELOCIDAD REGULADA DESDE 0 HASTA 255 o 0 hasta
-255

      ////////////////MOTOR IZQUIERDO/////////////

      if(izq>=0){

        digitalWrite(izq1, HIGH);

        digitalWrite(izq2, LOW);

      }

      else{

        digitalWrite (izq1, LOW);

        digitalWrite (izq2, HIGH);

        izq=izq*(-1);

      }

      analogWrite(pwmi,izq);



      ////////////////MOTOR DERECHO///////////////



       if(der>=0){
```

```
    digitalWrite(der1, HIGH);
    digitalWrite(der2, LOW);
   }
   else{
    digitalWrite (der1, LOW);
    digitalWrite (der2, HIGH);
    der=der*(-1);
   }
   analogWrite(pwmd,der);
}
```

Imagen 218. Programa para motores 1.

Imagen 219. Programa para motores 2.

Para probar el programa primero se debe compilar para verificar que no contenga errores, eso se hace en la parte superior del programa en la palomita. Una vez presionado ese botón se debe mostrar en la parte inferior la palabra “Compilado” posteriormente en herramientas se debe seleccionar la placa: “Arduino Nano”, el procesador: “ATMega328P (OldBootloader)” y por último el puerto: Este puede ser “COM9, COM10, COM11” puede variar. Para cargar el programa se debe tener conectado el Arduino Nano a la computadora por medio de USB y en caso de solo

ser el Arduino no hay problema, pero en caso de tener instalado el Arduino a la placa todos los Switch deben estar abajo o en su defecto la placa no debe estar energizada para evitar que el Arduino se queme. En la parte superior izquierda de la pantalla se visualizara una flecha a la derecha, ahí se debe dar click para cargar el programa. Con esto se pueden realizar pruebas a los motores sin problema.

5.2 Programación de motor con respecto al pwm

La finalidad del programa de los motores es prácticamente la modificación de la frecuencia del Timer que controla el PWM, en el cual están conectados los pines que emulan la salida analógica para el control de la velocidad de los motores, esto con el fin de que ambos motores vayan a la misma velocidad y tengan la misma reacción a la frecuencia que se seleccione dentro del código.

Recordando la información de los PWM (Página xx) se deben tomar en cuenta las frecuencias predeterminadas para cada pin de PWM del Arduino Nano, que son las siguientes:

Frecuencia PWM para D3 y D11: 490.20 Hz (El DEFAULT)

Frecuencia PWM para D5 y D6: 976.56 Hz (El DEFAULT)

Frecuencia PWM para D9 y D10: 490.20 Hz (El DEFAULT)

Los pines seleccionados para asignar el PWM en la placa fueron el D3 y D11 como se ha mencionado dentro de esta programación de motores, trabajando a 490.20 Hz donde la velocidad de los motores será regulada por esta selección de PWM, se descartaron los pines D5 y D6 debido a que estos pines utilizan el mismo TIMER que domina a los retardos, al DELAY dentro del Arduino, y los pines D9 y D10 no fueron utilizados debido al trabajo que se realizó con la turbina

en donde estos pines cumplen una mejor función con la turbina que con los motores y están pensados para trabajar con ella.

5.2.1 Porque no trabajar con pines analógicos de diferentes frecuencias.

Se ha hablado del porque se utilizan los pines D3 y D11, también se ha mencionado que es lo que sucede si se eligen pines diferentes, por tanto queda por aclarar lo que sucede en caso de trabajar con pines analógicos de diferentes frecuencias, por ejemplo: En un caso hipotético se selecciona trabajar con un pin D11 y un pin D6. En un principio se tiene para cada pin diferentes frecuencias, por un lado para el pin D11 se tiene una frecuencia de trabajo de 490.20 Hz y por otro lado para el pin D6 se tiene una frecuencia de trabajo de 976.56Hz, prácticamente el doble de frecuencia en cuestión al primer pin, hablando en cuestión de frecuencias por defecto, que son las que se presentan. Como resultado de estas variaciones de frecuencia se tendrá por consecuencia una diferencia de velocidad en los motores, presentando una velocidad mayor en el pin D11 y una velocidad menor en el pin D6, existiendo una inarmonía de trabajo entre ambos. Ocasionando una mala respuesta general, tanto en velocidad, giro, tramos rectos, curvos o esquinas que se presenten en pista. Para esto es importante conocer el funcionamiento el microcontrolador, el Arduino Nano trabaja con el microcontrolador ATMEGA328, para conocer sus capacidades se debe entender como está utilizando el Bootloader de Arduino y derivado de esto se puede modificar la frecuencia de los motores en base a la programación.

5.2.2 Programación de los motores con PWM

Tomando como base el primer programa para probar los motores, se agregaran las líneas de código correspondientes a la modificación de frecuencias del PWM, para ello se utilizara la información siguiente que es la frecuencia de PWM para pines D3 y D11

5.2.3 Frecuencia de pwm para pines d3 & d11

TCCR2B = TCCR2B & B11111000 | B00000001;

set timer 2 divisor to 1 for PWM frequency of 31372.55 Hz

TCCR2B = TCCR2B & B11111000 | B00000010;

set timer 2 divisor to 8 for PWM frequency of 3921.16 Hz

TCCR2B = TCCR2B & B11111000 | B00000011;

set timer 2 divisor to 32 for PWM frequency of 980.39 Hz

TCCR2B = TCCR2B & B11111000 | B00000100;

set timer 2 divisor to 64 for PWM frequency of 490.20 Hz (The DEFAULT)

TCCR2B = TCCR2B & B11111000 | B00000101;

set timer 2 divisor to 128 for PWM frequency of 245.10 Hz

TCCR2B = TCCR2B & B11111000 | B00000110;

set timer 2 divisor to 256 for PWM frequency of 122.55 Hz

TCCR2B = TCCR2B & B11111000 | B00000111;

set timer 2 divisor to 1024 for PWM frequency of 30.64 Hz

Esta información debe ser copia y pegada tal cual en el programa de los motores del inicio justamente debajo del void setup y se deben de ir colocado // a cada línea de código para hacerlas comentarios puesto que solo se trabajara con una de todas las líneas de código nuevas que se agregaron en base al PWM. Ahora se puede observar cómo se modifica la frecuencia dentro ID de

Arduino para el TIMER 2 el cual controla los pines D3 y D11 como salía analógica, en caso de dejar todas las líneas nuevas de código como comentarios, se tiene una frecuencia de trabajo ya establecida o por defecto que responde a 490.20 Hz que incluso se menciona delante de la línea de código (The DEFAULT), esta frecuencia es una frecuencia por defecto que sin necesidad de estas líneas de código esta activa para estos motores en base a los pines seleccionados, pero la finalidad de estas nuevas líneas de código es poder modificar la frecuencia aumentándola o reduciéndola según sea el caso. Cuando existe esta reducción o incremento de la frecuencia los motores trabajan mejor o peor. Para elegir una frecuencia lo único que se debe realizar es quitar los // del código para que no quede como comentario sino como línea de código activa, en caso de querer una línea de código menor se elige la cantidad de Hz menor, se quitan los // y listo, también es importante verificar que las demás líneas queden como comentario. Y este proceso debe ser para cualquier línea de código que se quiera activa.

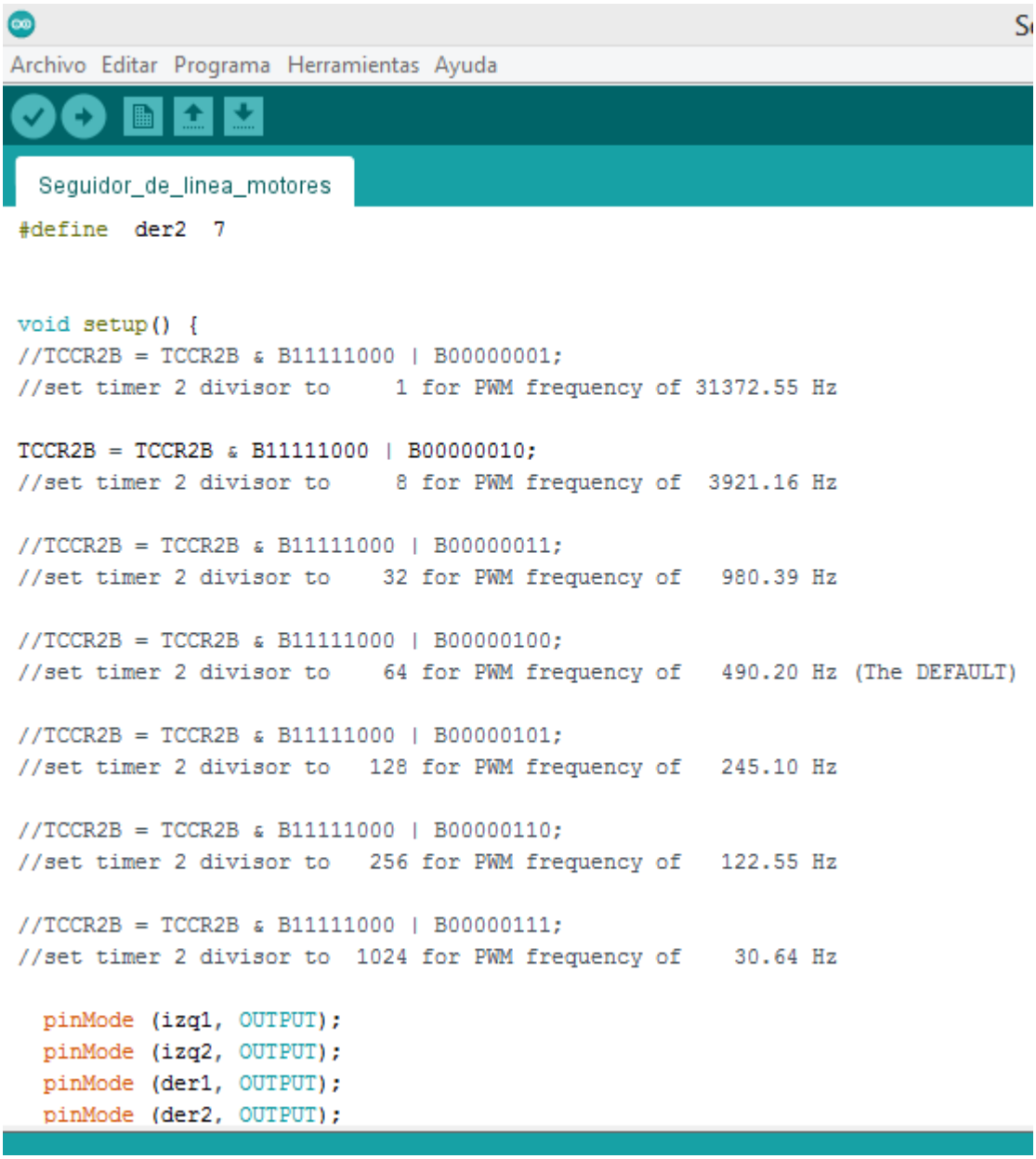


Imagen 220. Programa para motores con respecto a la frecuencia del PWM.

5.2.4 ¿En qué momento se ve reflejado este cambio de frecuencias?

El cambio de frecuencias se ve reflejado cuando el programa es cargado al seguidor y es activado. Dependiendo de la velocidad a la cual el motor es sometido a prueba estos cambios pueden ser más notorios, en cuanto a visión y sonido. En primera, utilizando la frecuencia por defecto (490.20 Hz) se puede escuchar como los motores del seguidor tal motor de auto tienen un sonido particular, en donde pueden escucharse con un mayor esfuerzo, un mayor volumen, etc. Y lo segundo es en cuanto a visión, puesto que la oscilación del motor puede ser notoria a una velocidad baja. Este par de efectos pueden ser aún más notorios con la frecuencia más baja para los motores, esto es que al utilizar una frecuencia de 30.40 Hz su sonido será aún más notorio, más fuerte y de menor calidad por decirlo de algún modo, y la oscilación será también mayor. Además de la selección de frecuencia otro factor importante que define las alteraciones en el giro es el Driver y los motores como tal.

5.2.5 La mejor frecuencia para los motores

El hecho de subir la frecuencia al máximo no es sinónimo de una mejor respuesta para los motores, sin bien es claro que teniendo la frecuencia al máximo (31372.55 Hz) se tendrá una respuesta real en cuanto al desempeño de los motores, también es importante resaltar que la velocidad en los motores no será la mejor, o las más rápida.

Para esto se recomienda utilizar entre 3921.16 Hz y 980.39 Hz obteniendo mejores resultados en rectas, mejor aceleración en curvas, en esquinas o ángulos de 90°, una respuesta mayor a la frecuencia por defecto. Pero sobre todo, la mejor frecuencia para los motores debe ser seleccionada en base a pruebas con el seguidor de línea sobre pista. Las frecuencias mencionadas son las más adecuadas en cuestión de motores tipo Pololu.

5.3 Calibración de la barra de sensores

La programación efectuada para la calibración de la barra multiplexada de 16 sensores es para una barra de Ingeniero Maker, pero este mismo programa puede ser utilizado para una barra de Phoenix o similar con tan solo un cambio en el código establecido que se explica posterior a la realizar del programa. Ambas regletas utilizan un multiplexor CD74HC4067 el cual es un multiplexor de 16 canales a 1, que permite a la entrada del multiplexor 16 canales, y a la salida 1 la cual es la que se va a lecturar, dependiendo de la combinación lógica a utilizar para poder lecturar cada uno de los sensores. La siguiente tabla de verdad se encuentra en el DataSheet del multiplexor CD74HC4067 la cual resulta ser muy importante para el código del programa debido a que con ella se sabrán los pines a utilizar para realizar las combinaciones lógicas y seleccionar un canal para realizar la lectura de uno de los sensores.

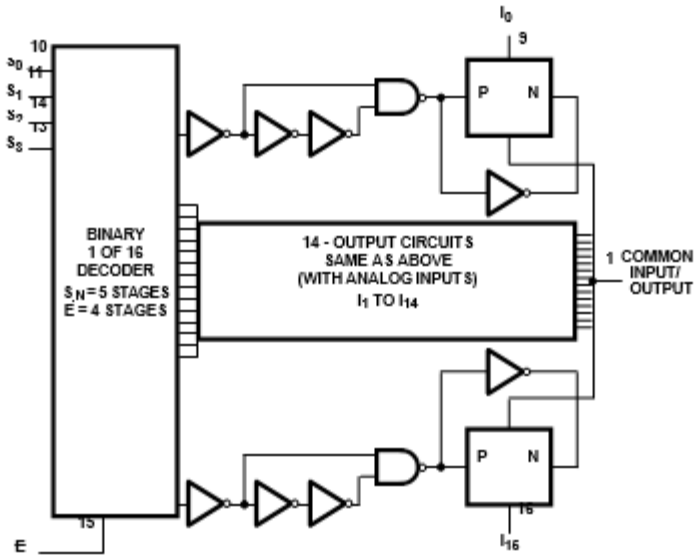
En la tabla de verdad se puede observar su distribución teniendo de inicio a las entradas S0, S1, S2 Y S3, seguidos de una entrada Enable Negada (E) y la selección de canal. La entrada Enable es una entrada de habilitación la cual esta negada lo que significa que se habilita con un 0 lógico y por tanto si se coloca un 1 lógico en esa entrada Enable no se tendría ninguna lectura, como consecuencia no se tendría seleccionado ningún canal. Muestra de ello es la primera fila donde se encuentra en todas las entradas S una X y en E un 1, dando por resultado una no lectura, y ningún canal.

El pin Enable ya se encuentra conectado a 0 Volts en la regleta, asegurando solo colocar la combinación binaria para poder seleccionar uno de los canales.

La regleta de Ingeniero Maker tiene los pines con el mismo orden de la regleta QTR8 de Pololu, y de la cual se tomó como base su orden para el diseño de la placa. El orden es el siguiente de izquierda a derecha:

OM, SL3, SL2, SL1, SL0, LON, +5 y GND

Functional Diagram



TRUTH TABLE

S0	S1	S2	S3	$\overline{E}$	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

H= High Level
L= Low Level
X= Don't Care

Imagen 221. Tabla de verdad del multiplexor CD74HC4067.

Imagen 222. Barra de sensores multiplexada de Ingeniero Maker.

Con este diseño definido se permite el uso tanto de la regleta de Maker, de Phoniex, de Pololu, clones y en teoría casi cualquier tipo de regleta en general, siendo el funcionamiento de la placa de forma genérica la que permite la configuración para usar con diferentes barras de sensores. Teniendo el siguiente orden en conexión de la placa para la barra de sensores:

A7, A6, A5, A4, A3, A2, A1, A0, Led ON, VCC, GND

Teniendo a los pines del A7 al A0 como entradas analógicas, y como salidas digitales a los pines A3, A2, A1, A0 que son las señales que se necesitan combinar digitalmente para poder seleccionar uno de los canales y poder lecturarlo por el pin A4.

Comparando la configuración de las salidas de la placa con las de la regleta de sensores de Ingeniero Maker se tendría lo siguiente:

OM, SL3, SL2, SL1, SL0, LON, +5, GND

A4, A3, A2, A1, A0, Led ON, VCC, GND

Se puede observar que las conexiones son equiparables y pensadas en tener congruencia entre ellas, cabe mencionar que de la placa solo se utilizara la conexión hasta la entrada A4 dejando a las entradas A7, A6 y A5 libres.

Se mencionan y comparan las entradas o pines entre la placa y la regleta porque así serán las conexiones físicas entre la placa y la regleta, para una regleta de Ingeniero Maker tal como se mostró en la comparación anterior así son las conexiones. Para una regleta diferente se tendrían que tomar en cuenta los pines de su regleta con los de la placa y realizar la conexión que será en un orden distinto y solo se tendrían que conectar con su homónimo, caso diferente a este, en que ya fue pensada una conexión directa.

Regresando a la información presentada en la tabla de verdad del DataSheet se indica que si todos los pines de entrada digital en el multiplexor se encuentran apagados, como resultado se seleccionaría el canal 0 por tanto se estaría lecturando el primer sensor de la regleta, En caso de tener únicamente el primer pin de entrada digital de la regleta del multiplexor se encontrara habilitado y los otros 3 deshabilitados, se seleccionaría el siguiente sensor, el canal 1. Y así sucesivamente con todos los canales restantes.

Entonces es simple, lo único que se tiene que hacer para el programa seria declarar los pines de entrada digital (S0, S1, S2 Y S3), realizar la lectura de datos, y posteriormente haciendo un barrido se podrían lecturar todos al mismo tiempo.

5.4 Programación para la lectura de la barra de sensores

Para iniciar con la programación se debe abrir un nuevo proyecto de Arduino y comenzar por declarar los pines que se indicaron, también se tienen que definir los nombres de cada uno de los pines a los cuales está conectada la barra de sensores. Iniciando con #define s0 14. 14 es equivalente al pin digital A0, y como se mencionó anteriormente también el pin A1 funciona como pin digital 15, y así sucesivamente hasta el pin A5 que es el equivalente al pin digital 19 dentro del Arduino que se pueden utilizar como salida y entrada digital y de igual forma como entrada analógica.

Situación que no es necesaria más que para el pin A4 que se ocupara como entrada analógica y del pin A0 a A3 como salidas digitales. Una vez que se tienen estos pines ya designados se procede a realizar el código para S1, S2 y S3. Ahora con las salidas digitales ya establecidas para el programa, inmediatamente a ellas se debe declarar la entrada analógica como AI (Analog In) conectado al pin A4. Anexo a las declaraciones de variables realizadas también se declarara el LED con el pin 13, LED que el Arduino tiene conectado de forma interna en la placa para poder observar la calibración de los sensores con respecto de la pista, además de este LED que indica la calibración de la barra de sensores, también se debe declarar el LED_ON encargado de habilitar los sensores para el pin 12 y por último se debe definir el botón, para el pin 9. Dando por resultado el siguiente código:

```
#define S0              14
#define S1              15
#define S2              16
#define S3              17
#define AI              A4
#define LED_ON     12
#define LED             13
#define BOTON        9
Int sensores [16];
```

Lo siguiente es integrar una variable de tipo entero ya que se va a realizar una lectura de 0 a 1023 que son 1024 posiciones y combinaciones posibles de procesamiento de lectura analógica que tiene el Arduino, con la variable int sensores [16]; (Arreglo de 16 posiciones).

Para el void setup se tienen que definir las salidas digitales S0, S1, S2 y S3 pero sin declarar una entrada analógica debido a que las entradas analógicas se declaran por defecto cuando se utiliza la lectura analógica, además se tienen que definir los pines digitales de LED_ON y LED además de la entrada digital del botón pin 9, el código seria el siguiente:

```
pinMode (s0, OUTPUT);

pinMode (s1, OUTPUT);

pinMode (s2, OUTPUT);

pinMode (s3, OUTPUT);

pinMode (LED_ON, OUTPUT);

pinMode (LED, OUTPUT);

pinMode (BOTON, INPUT);
```

Para comprobar que la teoría del multiplexor es cierta se debe realizar una combinación digital de lo pines s0, s1, s2 y s3 para lecturar uno de los sensores, para ello en void loop se realiza con ayuda de la variable digitalWrite (s0, 0) y se va a apagar, la combinación con todos los pines de la siguiente manera:

```
digitalWrite (s0, 0); digitalWrite (s1, 0); digitalWrite (s2, 0); digitalWrite (s3, 0);
```

Prestando atención a esta combinación de código se puede comparar y tomar como referencia a la tabla de verdad del DataSheet del multiplexor teniendo la primer combinación indicada en la tabla de verdad donde están apagados todos los pines de salida digital, y por tanto en la selección de canal se tiene al canal 0 que corresponde a uno de los sensores del costado de la barra. No se puede indicar cual sensor es del que se habla, ciertamente es un sensor ubicado a los laterales de la barra de sensores pero se desconoce hasta este momento cual, por lo tanto primero se debe realizar la lectura del programa y ver cuál es el sensor que arroja lectura. Para realizar la lectura nuevamente en el programa de Arduino se agrega a la última línea de código lo siguiente: sensores [0] =analogRead (AI); para así terminar con la línea de código de la siguiente forma:

digitalWrite(s0, 0);digitalWrite(s1, 0);digitalWrite(s2, 0);digitalWrite(s3, 0);sensores[0]= analogRead (AI);

Esta línea tiene la función de realizar la lectura para los sensores, más específicamente para el sensor 0, en la posición 0 del vector que tiene 16 posiciones. Lo que sucede es que se está cargando el dato de lectura analógica a la posición 0 de la variable sensores, y al final solo queda imprimir la variable con la siguiente línea de código:

Serial.println(sensores[0]);

También, después del void setup se debe habilitar la comunicación Serial a una velocidad de 115200 Vaults con el siguiente código:

Serial.begin(115200);

Y habilitar el LED ON que indicará que el seguidor está conectado y que los procesos están en función, con la siguiente línea de código debajo de la línea de BOTON:

digitalWrite(LED_ON, 1);

El programa hasta este preciso momento ha quedado de la siguiente manera:

```
#define s0        14

#define s1        15

#define s2        16

#define s3        17

#define AI        A4

#define LED_ON    12

#define LED       13

#define BOTON     9

int sensores [16];


void setup() {

Serial.begin (115200);

pinMode (s0,OUTPUT);

pinMode (s1,OUTPUT);

pinMode (s2,OUTPUT);

pinMode (s3,OUTPUT);
```

```
    pinMode (LED_ON,OUTPUT);
    pinMode (LED,OUTPUT);
    pinMode (BOTON,INPUT);
    digitalWrite(LED_ON,1);
    }

    void loop() {
    digitalWrite(s0,0);digitalWrite(s1, 0);digitalWrite(s2, 0);digitalWrite(s3,
0);sensores[0]=analogRead(AI);
    Serial.println(sensores[0]);
    }
```

```
#define s0      14
#define s1      15
#define s2      16
#define s3      17
#define AI      A4
#define LED_ON  12
#define LED     13
#define BOTON    9
int sensores[16];

void setup() {
Serial.begin(115200);
pinMode(s0,OUTPUT);
pinMode(s1,OUTPUT);
pinMode(s2,OUTPUT);
pinMode(s3,OUTPUT);
pinMode(LED,OUTPUT);
pinMode(LED_ON,OUTPUT);
pinMode(BOTON,INPUT);
digitalWrite(LED_ON,1);
}

void loop() {

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[0]=analogRead(AI);
Serial.println(sensores[0]);
```

Imagen 223. Lectura de sensores, sensor 0.

Lo que resta por realizar es compilar el programa con la opción de verificar que se encuentra en la parte superior izquierda con un icono de paloma y que una vez compilado en la parte inferior izquierda se observe la leyenda “Compilado”, se debe cargar el programa al Arduino iniciando por tener conectado el Arduino a la computador por medio de USB, haber seleccionado los puertos de comunicación, el procesador y la placa (se menciona en el programa para motores) y dando click a la flecha ubicada en la parte superior izquierda a un costado de la opción “Verificar” la cual compila el programa, para poder cargarlo.

Una vez cargado el programa en el seguidor, para poder comprobar cuál es el sensor 0 en la regleta, se debe colocar el seguidor sobre una hoja blanca, con una flecha negra sobre la hoja, esta flecha y hoja nos permitirán tener un fondo y una línea haciendo alusión a una pista para el seguidor de línea velocista, y sobre esta hoja se debe mover al seguidor haciendo que la regleta pase entre

la línea central de la flecha de izquierda a derecha para identificar cuál de los sensores de los costados es el sensor 0.

Para esto en la parte superior derecha de Arduino se encuentra una lupa que corresponde al Monitor Serial el cual permitirá observar el comportamiento de los sensores.

En un principio se observarán valores en forma de lista, los cuales puede ser de 150 a 180 aproximadamente, y esos valores subirán o bajaran dependiendo de la posición del sensor que se está localizando, por tanto es importante pasar sensor por sensor sobre la línea negra, enfocado principalmente a los sensores de los costados, para identificar el sensor 0. Como respuesta correcta a la ubicación del sensor, lo que sucederá es que el valor que se muestre en el monitor Serial será de 955 aproximadamente, mostrando un aumento en el valor pasando de un valor menor en el fondo blanco para los sensores que no se seleccionaron a un valor mayor para el sensor que se buscaba, dando como resultado el sensor derecho de la barra de sensores como el sensor buscado (Sensor 1).

Ahora para comprobar el ultimo sensor, o sea el sensor 15 (16 para la barra de 16 sensores) se debe colocar a todas las entradas “S” en un 1 lógico, para que tal como lo muestra la tabla de verdad, se seleccione el canal 15, y por ende se tenga lectura en el sensor 16, el sensor ubicado en el extremo contrario de la barra de sensores, por tanto lo único a realizar es colocar 1, para s0, s1, s2 y s3 en el código, en la posición y la impresión del sensor, en vez de 0 tener 15 quedando de la siguiente manera:

```
digitalWrite(s0,1);digitalWrite(s1, 1);digitalWrite(s2, 1);digitalWrite(s3, 1);sensores[15]=analogRead(AI);

Serial.println(sensores[15]);
```

Siendo el único cambio realizado, nuevamente se compila el programa, se carga y se realiza la misma prueba sobre la hoja blanca, enfocándonos en el sensor del extremo contrario al sensor 0, donde de igual manera se puede revisar sensor por sensor para ver su comportamiento numérico en el monitor Serial.

Este procedimiento se puede repetir con cualquiera de los sensores, tanto para extremos como para sensores internos, lo único que se debe hacer es consultar la tabla de verdad, elegir el sensor a poner a prueba, colocar los valores de las entras de “S” y cambiar la posición y la impresión del sensor a revisar.

Como se ha probado qué sensor inicia, por lógica se puede comenzar contando por ese sensor para identificar todos los demás. La lectura de los sensores puede variar dependiendo de diferentes factores, tales como la luz y la altura a la que se encuentre la barra de sensores de la pista, para el caso de las regletas de Ingeniero Maker se recomienda tener entre aproximadamente 3 a 5 mm de altura para la barra de sensores.

Este orden de los sensores puede ser diferente para otras regletas, se podrían tener los sensores 0 y 15 en extremos contrarios a los que presenta la regleta de Ingeniero Maker, y no solo eso, sino que al momento de revisar cada sensor, la respuesta puede ser distinta, en este caso puede que cuando el sensor este ubicado en el fondo de la pista el valor que arroje para el monitor Serial sea un valor alto, y que al colocar dicho sensor sobre la línea negra de la pista el monitor Serial arroje un valor bajo. Lógicamente se tendrían que realizar pruebas con regletas distintas a la utilizada en el proyecto para conocer todos estos parámetros. Para este caso se utilizó una regleta distinta a la de Ingeniero Maker, para ser concretos de Phoenix y se pudo notar que se presentaron estas características mencionadas, por tanto se realizó una modificación al código para que pueda ser

utilizado con la placa del proyecto, con el mismo programa pero cambiando una mínima de valores para pueda funcionar con una barra de sensores distinta. Ese cambio en el código se explicara más adelante, por el momento el programa seguirá enfocado a la lectura de los sensores.

Para continuar con el programa se copiara la línea de código para la selección de los sensores basado en la tabla de verdad, y se pegara 15 veces más para tener el total de 16 canales para la selección de los sensores, la línea de código es la siguiente:

digitalWrite(s0,0);digitalWrite(s1, 0);digitalWrite(s2, 0);digitalWrite(s3, 0);sensores[0]=analogRead(AI);

Lo único que se debe cambiar es el valor de las entradas "S" en cada línea de código, pero con el orden de la tabla de verdad del multiplexor, tal cual, uno tras otro, para tener los canales del 0 al 15 de la siguiente manera:

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[0]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[1]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[2]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[3]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[4]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[5]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[6]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[7]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[8]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[9]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[10]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[11]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[12]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[13]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[14]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[15]=analog Read(AI);

```
void loop() {
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[0]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[1]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[2]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[3]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[4]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[5]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[6]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[7]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[8]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[9]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[10]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[11]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[12]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[13]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[14]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[15]=analogRead(AI);
Serial.println(sensores[15]);
}

Compilado
```

Imagen 224. Selección de canales del 0 al 15.

De esta manera se puede seleccionar entre uno u otro sensor, saltar de canal en canal de forma sencilla y así obtener sus lecturas, de esto modo no se pueden imprimir las lecturas debido a que ya se estarían realizando la lectura de todos los sensores al mismo tiempo ya que no se está utilizando ningún retardo pero internamente en el procesador son varias líneas de código utilizadas al momento de ejecutar todos los códigos que se agregaron para los demás sensores. Para poder realizar la impresión de todos los sensores es necesario utilizar un bucle “for” con un comienzo en 0 y lecturando los 16 sensores que tiene la regleta, sumando uno por uno, teniendo la siguiente línea de código:

```
for(int i=0;i<16;i++){
  Serial.print(sensores[i]);
  Serial.print("\t");
```

```
}
Serial.println("");
}
```

También se modifica la impresión de los sensores, y para la impresión de los sensores se cambiara a que se imprima línea por línea, entonces, en vez de imprimir al sensor 0 o al sensor 15, se cambia por la variable i para que el programa imprima sensor por sensor, iniciando por el sensor 0, sensor 1, sensor 2, etc, hasta llegar al sensor 15, 15 que es menor que 16, también se agregara un espacio de sensor a sensor, para que los datos no choquen entre sí con la siguiente línea de datos:

```
Serial.print("\t");
```

Y posteriormente a la impresión del sensor 15 se agregara una nueva línea de código para que salte un renglón, este será un espacio en blanco con un salto de línea para que todos los sensores se lecturen, cada sensor por cada fila de datos:

```
Serial.println("");
```

Todo este proceso realizado debe ser compilado nuevamente, y en el monitor Serial se podrá observar el resultado de la modificación realizada en las líneas de código, cambiando de tener solo una línea de datos de un sensor por 16 líneas de datos para todos los sensores de la regleta, cada uno con su respectiva lectura. Los sensores de los constados mostraran valores bajos para la regleta de Ingeniero Maker, mientras que los dos sensores del centro posicionados sobre la línea negra de la hoja deberán lecturar valores altos. Al momento de pasar sensor por sensor sobre la línea negra, el sensor en turno modificará su valor de bajo a alto.

Como se obtienen resultados de valores de tres dígitos y con una interpretación un tanto incierta una opción conveniente sería modificar estos valores a 0 y 1 únicamente, los cuales indicaran si el sensor esta sobre la línea o sobre el fondo, por ejemplo: En caso de tener un valor menor a 400 el sensor esta sobre fondo, por tanto el valor en vez de ser 400 se convertiría a 0, indicando que el sensor se encuentra en fondo y no en línea. En caso contrario de tener un valor mayor a 400 el sensor estaría sobre la línea lo cual se convertiría en 1, indicando que el sensor se encuentra en línea (valor hipotético de 400).

Para esto se agrega una línea de código con if en el cual se debe indicar un umbral, el umbral será de tipo entero y el valor de los sensores i en donde se encuentra la posición será 0, garantizando que el valor mientras se encuentre en el fondo blanco corresponda a este digito. El código es el siguiente:

```
If (sensores[i]<=umbral){
Sensores[i]=0;
}
```

Además de esta línea de código al principio también se tiene que declarar el valor del umbral, en este caso el umbral será de 600 y la línea de código, será la siguiente:

```
int umbral=600;
```

Entonces basados en la línea de código con if se tendrá que, si el valor del sensor es menor o igual a 600 el valor del sensor será 0, y esta línea de código deberá ser colocada antes de la impresión de los valores, para que al momento de imprimir el valor sea el modificado. El orden es el siguiente:

```
for(int i=0;i<16;i++){
  if(sensores[i]<=umbral){
    sensores[i]=0;
  }
  Serial.print(sensores[i]);
  Serial.print("\t");

}
Serial.println("");
}
```

Por tanto cuando los sensores estaban en el fondo blanco marcaban valores bajos, que en este caso ahora serán 0, pero para valores altos, los sensores registraran valores altos, como los de 900 y algo, aproximadamente. Para comprobarlo se debe compilar el programa y cargarlo, una vez realizada la carga del programa se puede probar colocando el seguidor sobre la pista y notando que para los sensores sobre el fondo blanco se tendrá un valor de 0, y para los sensores sobre la línea se tendrán valores altos. Se debe mover cada uno de los sensores sobre la línea negra para verificar que la programación de los sensores sea correcta, y los sensores funcionen de manera correcta.

Lo que se busca con este tipo de pruebas es que una vez comprobado el funcionamiento de los sensores y que la programación este correcta se pueda realizar una posición con promedio ponderado, que garantice para los sensores con niveles altos que el seguidor está yendo en línea

recta y que en caso de que el seguidor presente un desvió de la trayectoria se pueda corregir la dirección para volver a esa posición, cambiando la velocidad de los motores, con respecto a la posición del seguidor.

Todo el código que se utiliza para la selección de canal de los sensores es bastante extensa, lo cual ocasiona que la memoria RAM del microcontrolador se vea afectado. Pero se puede compactar con ayuda de una combinación digital lógica a partir de un bucle for, para ello lo primero a realizar es dejar en un comentario todas las líneas de código de la selección de canales con ayuda de un /* para iniciar y */ para finalizar y se debe escribir un bucle for para lecturar los valores, con una variable entera i que comience en 0 que sea menor a 16 y realice un barrido de bits con el siguiente código:

```
for(int i=0;i<16;i++){
digitalWrite(s0,i&0x01);
digitalWrite(s1,i&0x02);
digitalWrite(s2,i&0x04);
digitalWrite(s3,i&0x08);
sensores[i]=analogRead(AI);
if(sensores[i]<=umbral){
sensores[i]=0;
}
Serial.print(sensores[i]);
```

```
Serial.print("\t");

}

Serial.println(" ");
```

El código es prácticamente lo mismo, solo que ahora se han ahorrado líneas de código y beneficiando al Arduino y su funcionalidad. Todo aquello que se dejado en comentario no afecta al programa en absoluto y se puede comprobar compilando el programa y subiéndolo al seguidor para verificar por medio de la pista que las respuesta de los motores es la misma. Teniendo 0 para valores inferiores al umbral, pero no solo eso sino que también se puede tener el valor de 1 para valores mayores al umbral, agregando un else después del if y en este caso se aplicara un caso contrario al descrito para el umbral inferior a cero, provocando un 1 para un valor de umbral mayor a 1, la línea de código será la siguiente:

```
else{sensores[i]=1;}
```

Se compila y se sube el programa para observar el comportamiento de los sensores en el monitor serial.

Para modificar el color de la línea y el fondo de la pista se debe color una variable entera a la que se denominara como línea y que cuando tenga un valor de 0 sirva para una pista blanca con línea negra, y cuando sea 1 funcione para una pista negra con línea blanca. La línea de código será la siguiente:

```
//I.M. BLANCO=1 NEGRO=0

//OTRAS BARRAS BLANCO=0 NEGRO=1

int linea=0;
```

Además de esto se debe colocar unas líneas de código con condicionales if que en caso de que la línea sea igual a 0 se deberán cumplir las condiciones de if y else y de igual manera para un caso contrario. El código es el siguiente:

```
if(linea==0){if(sensores[i]<=umbral){sensores[i]=0;}else{sensores[i]=1;}}

if(linea==1){if(sensores[i]<=umbral){sensores[i]=1;}else{sensores[i]=0;}}
```

Para el primer caso tenemos una línea de color negro (0) en la que si la respuesta de los sensores es menor al umbral la respuesta de los sensores se convertirán a 0 y en caso contrario si la respuesta de los sensores es mayor al umbral la respuesta de los sensores se convertirán a 1. Para el segundo caso es lo contrario, si la línea es de color blanco (1) si la respuesta de los sensores es menor al umbral la respuesta de los sensores se convertirá en 1 y en caso contrario si la respuesta de los sensores es mayor al umbral la respuesta de los sensores se convertirán en 0.

Con esta última implementación del código fácilmente se podrá cambiar entre barras de sensores, tanto para las de Ingeniero Maker como para otras barras de sensores. Con sus respectivos resultados, únicamente cambiando entre 0 y 1, para una línea negra y una blanca, obviamente con un fondo blanco y un fondo negro, y esto se puede verificar compilando el programa, subiéndolo al seguidor, probándolo sobre la pista y revisando sus valores en el monitor Serial.

Programa completo para lectura de sensores:

```
#define s0      14

#define s1      15

#define s2      16

#define s3      17
```

```
#define AI        A4

#define LED_ON    12

#define LED       13

#define BOTON     9

int sensores [16];

int digital [16];

int umbral=600;


//COLOR DE LA LINEA SEGUN LA BARRA DE SENSORES//

//I.M. BLANCO=1 NEGRO=0

//OTRAS BARRAS BLANCO=0 NEGRO=1

int linea=0;


void setup() {

Serial.begin (115200);

pinMode (s0,OUTPUT);

pinMode (s1,OUTPUT);

pinMode (s2,OUTPUT);
```

```
    pinMode (s3,OUTPUT);
    pinMode (LED_ON,OUTPUT);
    pinMode (LED,OUTPUT);
    pinMode (BOTON,INPUT);
    digitalWrite(LED_ON,1);
    }

    void loop() {

    }

    int lectura(void) {

/*digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[0]=analogR
ead(AI);
    digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[1]=analog
Read(AI);
    digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[2]=analog
Read(AI);
```

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[3]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[4]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[5]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[6]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[7]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[8]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[9]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[10]=analog Read(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[11]=analog Read(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[12]=analog Read(AI);

```
    digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[13]=analog
Read(AI);
    digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[14]=analog
Read(AI);
    digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[15]=analog
Read(AI);*/


    /*for(int i=0;i<16;i++){
      if(sensores[i]<=umbral){
        sensores[i]=0;
      }
      Serial.print(sensores[i]);
      Serial.print("\t");
    }
    Serial.println(" ");*/


    for(int i=0;i<16;i++){
      digitalWrite(s0,i&0x01);
      digitalWrite(s1,i&0x02);
```

```
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  sensores[i]=analogRead(AI);
  if(linea==0){if(sensores[i]<=umbral){digital[i]=0;}else{digital[i]=1;}}
  if(linea==1){if(sensores[i]<=umbral){digital[i]=1;}else{digital[i]=0;}}
  Serial.print(sensores[i]);
  Serial.print("\t");
}
Serial.println(" ");
}
```

```
#define s0      14
#define s1      15
#define s2      16
#define s3      17
#define AI      A4
#define LED_ON  12
#define LED     13
#define BOTON    9
int sensores[16];
int umbral=600;

//COLOR DE LINEA SEGUN LA BARRA DE SENSORES//
//I.M BLANCO=1 NEGRO=0
//OTRAS BARRAS BLANCO=0 NEGRO=1
int linea=0;

void setup() {
Serial.begin(115200);
pinMode(s0,OUTPUT);
pinMode(s1,OUTPUT);
pinMode(s2,OUTPUT);
pinMode(s3,OUTPUT);
pinMode(LED,OUTPUT);
pinMode(LED_ON,OUTPUT);
pinMode(BOTON,INPUT);
digitalWrite(LED_ON,1);
}

void loop() {
```

Imagen 225. Programa final, lectura de sensores parte 1.

```
void loop() {

/*digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[0]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[1]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[2]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[3]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[4]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[5]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[6]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[7]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[8]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[9]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[10]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[11]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[12]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[13]=analogRead(AI);
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[14]=analogRead(AI);
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[15]=analogRead(AI);*/
for(int i=0;i<16;i++){
  digitalWrite(s0,i&0x01);
  digitalWrite(s1,i&0x02);
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  sensores[i]=analogRead(AI);
  if(linea==0){if(sensores[i]<=umbral){sensores[i]=0;}else{sensores[i]=1;}}
  if(linea==1){if(sensores[i]<=umbral){sensores[i]=1;}else{sensores[i]=0;}}
  Serial.print(sensores[i]);
  Serial.print("\t");


}
Serial.println(" ");

/*for(int i=0;i<16;i++){
  if(sensores[i]<=umbral){
    sensores[i]=0;
  }
  Serial.print(sensores[i]);
  Serial.print("\t");
}
Serial.println(" ");*/

}
```

Imagen 226. Programa final, lectura de sensores parte 2.

5.5 Calibración de sensores

La calibración de los sensores es de suma importancia debido a que como se ha visto en el programa, para la lectura de los sensores se ha utilizado un valor de 600 para el umbral, siendo esta cifra una cifra al alzar, un valor propuesto por nosotros de forma empírica, a prueba y error. Como resultado de este valor de umbral aleatorio el programa puede definir como 1 o 0 lógico al nivel registrado por los sensores en base al umbral.

Con la calibración de los sensores se podrá definir un valor de umbral específico para el programa, un valor con el que el PID podrá realizar las correcciones de error, controlar la velocidad de ambos motores simultáneamente y podrá centrarse con el set point ubicada en el centro de la barra de sensores del seguidor de línea.

Para iniciar con la calibración de los sensores lo primero es abrir el programa para la lectura de la barra de sensores, y en la parte de void loop lo siguiente a realizar será seleccionar y cortar todas las líneas de código que se habían ingresado para posteriormente crear una nueva función con retorno de tipo entero la cual se denominara como “lectura”, entonces con ayuda de la variable int se colocara la siguiente línea de código: int lectura(void){

}

Debajo de la llave que cierra al void loop y justo en esta nueva línea de código se pegara las líneas de código que se copiaron, juntando todo lo que este en comentario.

```
int lectura(void) {
  /*digitalWrite(s0,0);digitalWrite(s1,0);digitalWr
digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(
digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(
digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(
digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(
```

Imagen 227. Arreglo “lectura”.

Partiendo del código para convertir los valores a 1 y 0 lógico se creara un nuevo arreglo de datos que se llama “digital” debido a que el valor analógico será utilizado en otro momento, por tanto se debe establecer la variable al principio del programa justo debajo del arreglo de sensores, se debe colocar: int digital[16];

```
int sensores [16];
int digital [16];
int umbral=600;
```

Imagen 228. Arreglo “digital” de 16 posiciones.

Por tanto se debe cambiar el arreglo en el código de conversión de 1 y 0 lógico, usando el arreglo “sensores” en vez de “digital”, ese mismo cambio también aplica para la impresión del resultado.

```
if(linea==0){if(sensores[i]<=umbral){digital[i]=0;}else{digital[i]=1;}}
if(linea==1){if(sensores[i]<=umbral){digital[i]=1;}else{digital[i]=0;}}
Serial.print(digital[i]);
Serial.print("\t");
```

Imagen 229. Cambio de arreglo en el código.

Ahora se deben crear dos nuevos arreglos, el primero será denominado fondo y el segundo línea, y serán colocados entre el void loop e int lectura con el siguiente código: void fondos (){ } void lineas (){ }

Y para estos dos nuevos arreglos se les debe colocar el código para lectura de sensores que se encuentra en el arreglo “lectura” con la diferencia de sensores, por lectura_fondo, para que esta función pueda realizar la lectura de los sensores, y a la vez se pueda guardar en esa misma variable, para la impresión también se debe hacer el cambio de sensores, por lectura_fondo, además como no se necesita convertir ningún dato, las condicionales if se pueden eliminar.

```
void loop() {

}
void fondos() {
  for(int i=0;i<16;i++){
  digitalWrite(s0,i&0x01);
  digitalWrite(s1,i&0x02);
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  lectura_fondo[i]=analogRead(AI);
  Serial.print(lectura_fondo[i]);
  Serial.print("\t");
}
Serial.println(" ");

}
```

Imagen 230. Void fondos.

Lo mismo sucede para el arreglo “líneas” con la diferencia de que la lectura no será para fondo sino para la variable línea.

```
void lineas() {
   for(int i=0;i<16;i++){
  digitalWrite(s0,i&0x01);
  digitalWrite(s1,i&0x02);
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  lectura_linea[i]=analogRead(AI);
  Serial.print(lectura_linea[i]);
  Serial.print("\t");
}
Serial.println(" ");

}
```

Imagen 231. Void líneas.

De esta manera se han creado dos funciones con una tarea específica, para el fondo la función únicamente guarda las variables de cada uno de los sensores que realizaron la lectura al fondo, para la función línea guardará todos los valores de los sensores lecturados en el arreglo, tal como con el fondo.

5.6 Calibración del seguidor

Para realizar la calibración lo que se busca es que por medio del botón, al colocar el seguidor de línea sobre la pista el primer paso que se realice sea el de la calibración de los sensores, por tanto al colocar el seguidor, la barra de sensores deberá estar colocada en su totalidad en el fondo, y con el botón al presionarlo el seguidor tomara lecturas para los sensores del fondo, de igual manera sucederá con la línea, para este caso la regleta debe ser colocada en su totalidad sobre la línea negra para después presionar el botón y que se realice la calibración de la barra de sensores respecto a la línea y viceversa en cuanto a un fondo negro con línea blanca. (Anexar imagen de ambas escenas de calibración).

Pero para que estas acciones funcionen de forma correcta es necesario realizar un promedio entre el fondo y la línea, que pueda reemplazar al umbral no definido con exactitud.

Para esto se debe crear un nuevo arreglo denominado “promedio” únicamente copiando el código de fondos o líneas y pegándolo en este nuevo arreglo pero eliminando la combinación digital, además se debe agregar una función umbral por cada una de las 16 posiciones que sea igual a la lectura del fondo + la lectura de la línea y para sacar el promedio se debe dividir el resultado de la suma de estos dos valores, en este caso no solo son 2 valores, debido a que cada arreglo tiene 16 posiciones será la suma de cada uno de los sensores calibrados en el fondo, más los valores calibrados en la línea y divididos entre dos para tener el valor del umbral, por último se debe imprimir el umbral.

```
}
void promedio(){
  for(int i=0;i<16;i++){
  umbral[i]=(lectura_fondo[i]+lectura_linea[i])/2;
  Serial.print(umbral[i]);
  Serial.print("\t");
}
Serial.println(" ");
}
```

Imagen 232. Void promedio.

Como se han creado 3 arreglos, estos mismos se deben declarar al principio del programa, tal como se hizo para “sensores” o para “digital” de esta manera:

```
#define BOTON       9
int sensores [16];
int digital [16];
int lectura_fondo[16];
int lectura_linea[16];
int umbral[16];
//int umbral=600;
```

Imagen 233. Declarando arreglos.

Tenemos los arreglos: lectura_fondo, lectura_linea, y umbral, como tal no se coloca promedio sino umbral, y para el arreglo de umbral al que se le coloco un valor, únicamente se deja como comentario, puesto que el umbral ya se está calculando en base al promedio, por tal motivo un umbral definido por nosotros ya no es necesario. Este cálculo para el umbral permitirá que cada sensor tenga un umbral propio para que se pueda determinar el código sensor por sensor y permita realizar el cambio de 1 y 0 por cada sensor.

Lo que resta dentro de lectura es utilizar la función promedio para poder cambiar la lectura a digital, para el arreglo lectura en las condicionales if a umbral se le agregara [i], de este modo se

realiza la comparación por cada sensor, uno por uno, así se garantiza que el umbral es único para cada sensor y se tiene una calibración totalmente eficiente.

```
sensores[i]=analogRead(AI);
if(linea==0){if(sensores[i]<=umbral[i]){digital[i]=0;}else{digital[i]=1;}}
if(linea==1){if(sensores[i]<=umbral[i]){digital[i]=1;}else{digital[i]=0;}}
Serial.print(digital[i]);
Serial.print("\t");
```

Imagen 234. [i] para umbral .

Una vez realizada la modificación para el umbral lo siguiente es que mediante los valores digitales que se están capturando se realice el promedio ponderado a cada uno de los sensores con valores de 100 en 100, partiendo de 1500 para el primer sensor hasta 0 para el último, creando así un nuevo arreglo al que se denomina como "suma ponderada". Este promedio ponderado será la suma de todos los valores multiplicados por una ponderación, dividido entre la suma de todos estos valores, para así obtener en que rango de esos valores se encuentra el resultado (posición).

$$\bar{X} = \frac{\sum_{i=0}^{15} d_i * w_i}{\sum_{i=0}^{15} d_i} = \frac{(d_{15} * 1500) + (d_{14} * 1400) + (d_{13} * 1300) + \cdots (d_0 + 0)}{d_0 + d_1 + d_2 + \cdots d_{15}}$$

Donde:

D=datos digitales del código

W=ponderación de 0 a 1500 "100 por cada sensor"

Tres nuevas variables de tipo entero son necesarias para la suma ponderada, serán definidas al principio del programa de la siguiente manera: long int sump, suma, pos; para este caso se tendrá la suma pondera, la suma y la posición.

```
int umbral[16];
//int umbral=600;
long int sumap, suma, pos, position;
```

Imagen 235. Suma ponderada, suma y posición.

Y nuevamente en lectura se realizará la suma ponderada teniendo un valor ponderado para cada uno de los sensores, en este caso se va a colocar a cada uno de los valores el inverso en ponderación, por ejemplo: 1500*digital[15] sumado al siguiente sensor de esta manera:

sumap=(1500*digital[15]+1400*digital[14]+1300*digital[13]+1200*digital[12]+1100*digital[11]+1000*digital[10]+900*digital[9]+800*digital[8]+700*digital[7]+600*digital[6]+500*digital[5]+400*digital[4]+300*digital[3]+200*digital[2]+100*digital[1]+0*digital[0]);

Ahora que esta lista la suma ponderada de cada sensor es necesario realizar la suma general que básicamente es lo mismo pero esta vez eliminando la ponderación de cada uno de los sensores, de la siguiente manera:

suma=(digital[0]+digital[1]+digital[2]+digital[3]+digital[4]+digital[5]+digital[6]+digital[7]+digital[8]+digital[9]+digital[10]+digital[11]+digital[12]+digital[13]+digital[14]+digital[15]);

Lo único que resta por hacer es calcular el promedio ponderado, la cual es la posición, entonces se debe colocar la posición = a la suma ponderada/suma con el siguiente código:

pos=sumap/suma;

Y como la posición ya se debe devolver también se debe colocar un retorno para la posición desde esta función lectura: return pos;

Se debe llamar a cada función desde el void setup para que todos los datos se guarden paulatinamente, debido a que el botón ya está declarado como entrada, ahora se debe realizar un

bucle que impida la salida de información mientras no se presione el botón, en este caso conectado al pin 9: while(digitalRead(BOTON));

Este bucle impedirá la conexión del botón sin presionar, que va directamente a un 1 lógico, quedándose dentro del bucle hasta presionar el botón, por tanto como el LED esta encendido se quedara así hasta que se presione el botón para iniciar con la calibración, que como se mencionó antes primero se inicia por el fondo y después por la línea. Después se agrega un ciclo for para este proceso

```
digitalWrite(LED_ON,1);
digitalWrite(LED,1);
while(digitalRead(BOTON));
for(int i=0;i<50;i++){
  fondos();
  digitalWrite(LED,0);
  delay(20);
  digitalWrite(LED,1);
  delay(20);
```

Imagen 236. While y For para LED en calibración del fondo.

Se realizaran 50 iteraciones de esta pequeña función encendiendo y apagando el LED indicando que se está calibrando el fondo de la pista. Este código se repite para la línea de la pista, y la función hace que una vez que se presiona el botón el led parpadee indicando que se está calibrando con respecto del fondo, cuando el LED termine de parpadear significara que ha finalizado la calibración del fondo, posteriormente se agrega este código para que se repita el proceso pero con respecto a la línea.

```
while(digitalRead(BOTON));
for(int i=0;i<50;i++){
  lineas();
  digitalWrite(LED,0);
  delay(20);
  digitalWrite(LED,1);
  delay(20);
}
```

Imagen 237. While y For para LED en calibración de la línea.

Una vez que ha realizado la calibración en la línea, lo ideal es que se quede encendido el LED hasta presionar nuevamente el botón calculando antes el valor del promedio. Y una vez realizado este proceso se apagara el LED y después de apagarlo saldrá del void setup debido a que ninguna instrucción lo detiene, enviándonos al void loop

```
promedio();
while(digitalRead(BOTON));
digitalWrite(LED,0);
}
```

Imagen 238. While y For para LED en promedio.

En el void loop se tiene que introducir una variable que lea la posición, en este caso la función de lectura, esta variable deberá ser colocada con las otras 3 de sumap, suma, y pos de la siguiente manera:

long int sumap, suma, pos, position;

Esta variable será la encargada de leer la función lectura, que como retorno tiene la posición directamente. Lo que resta por hacer es imprimir cada línea con un salto de línea a la variable posición.

```
void loop() {
  position=lectura();
Serial.println(position);

}
```

Imagen 239. Posición.

Para que no exista confusión cuando se imprima línea tras línea se dejara como comentario el salto de línea ubicado antes del código para la suma ponderada, generando como resultado la entrega de cada valor digital de cada uno de los sensores sin obtener a un lado el valor de la posición.

```
,
//Serial.println(" ");
sumap=(1500*digital[0]+1400*digital[1]+1300*d
```

Imagen 240. Salto de línea como comentario.

Se continúa con la conexión del seguidor o el Arduino a la computadora por medio de USB para posteriormente seguir con la compilación del programa y su respectiva carga al seguidor, por medio del Arduino, puerto y procesador correspondientes. Una vez cargado el código al seguidor se puede iniciar con las pruebas de calibración, iniciando por el fondo, tal como se colocó en el código, para después proceder con la línea, para este caso al programa se le designo el 0, para la línea negra con la barra de sensores de Ingeniero Maker, por tanto se debe probar en un fondo blanco con línea negra, en caso de ser otra barra de sensores con la misma línea y fondo se deberá cambiar a 1 en el programa. Para observar el comportamiento de la calibración se debe abrir el monitor serial, entonces, al colocar el seguidor sobre el fondo, se debe presionar una vez el botón ubicado en la parte inferior izquierda del seguidor, para comenzar el proceso de calibración en fondo.

Se puede observar en el monitor serial cual es el comportamiento de respuesta de los sensores, que no debe variar mucho en un lugar cerrado, después cuando el LED del Arduino este encendido sin interrupciones se podrá proceder a realizar la siguiente calibración en la línea negra.

¿Qué sucede con los valores que están siendo arrojados en el monitor serial? bueno pues, para este caso primero se obtuvieron valores del fondo, y después de la línea, entonces el programa está sumando ambos valores y dividiéndolos, tal como se lo especificamos en el código y después en la última línea, la línea inferior de todos los valores presentados en el monitor serial se está obteniendo el valor del umbral, un valor de umbral para cada uno de los sensores, y un umbral real que proviene de cálculos realizador por el programa, y no un umbral designado de forma aleatoria.

Ya realizada la calibración tanto de fondo como de línea lo siguiente es obtener la posición, una siguiente línea a la derecha que debe aparecer con presionar una vez más el botón entregando el valor digital de cada uno de los sensores, del 0 al 15, y entregando a un costado derecho la posición. En este caso como la posición que se coloco fue con un valor de 0 a 1500, la mitad de la posición del seguidor tiene que ser 750 y justo ese valor corresponde al SET POINT en el cual se debe quedar el seguidor, ese valor será utilizado como la corrección de error para el parámetro proporcional.

Cuando el seguidor sale de la línea, es decir que los sensores reciben por respuesta única lecturas del fondo, el valor de posición tendrá como resultado -1, esto sucede debido a que no se está guardando el último valor de posición en la que se encontraba el seguidor, qué indica esto, que la lectura y la posición están correctos pero lo que falta es que el seguidor tenga en su memoria cual era la posición en la que se encontraba antes de salir de la línea.

El cambio se debe efectuar en el código, en la parte de variables long int donde se crearon las variables de suma ponderada, suma y posición, se debe crear una nueva variable que sea la de última posición: “poslast” esta será una variable que guarde la última posición antes de salir de la línea, y para que funcione esta variable, en la parte de lectura se deben crear un par de condiciones para que esto suceda. En este caso se usaran las condicionales if donde si la posición es menor o igual a 100 y si la posición es menor o igual a -1 entonces la posición actual deberá guardar el valor 0, garantizando una posición fija en caso de salirse de la línea. Esta misma línea de código servirá para cuando el seguidor salga por el lado contrario que sería un valor menor o igual a 1400 y si la posición es menor o igual a -1 entonces la posición actual deberá ser 1500. Por tanto no importa porque lado se salga, si sale por izquierda guardara el valor de 1500 y si es por la derecha guardara el valor de 0. Por último se tiene que hacer que se recuerde la posición anterior sea igual a posición dando como resultado las siguientes líneas de código:

```
pos=sumap/suma;
if(poslast<=100 && pos==-1){
  pos=0;
}
if(poslast>=1400 && pos==-1){
  pos=1500;
}
poslast=pos;
return pos;
}
```

Imagen 241. Guardado de la última posición.

Se compila y carga el programa nuevamente, y se realiza el proceso de calibración, iniciando por fondo, después línea y una vez más se presiona el botón para obtener la posición y poder observar el valor que arroja la lectura de sensores, cuando se sale el seguidor tanto por derecha como por izquierda, teniendo los valores de 0 y 1500 respectivamente para cada lado.

Lo que sucede cuando se utiliza una barra distinta a la de Ingeniero Maker es que el proceso es totalmente el mismo, incluso para la calibración tanto en fondo como en línea, la diferencia se encuentra en la posición cuando el seguidor sale por el lado izquierdo y el lado derecho, puesto que por el lado izquierdo guardara el valor de 0 y para la derecha 1500, y por tanto como esto puede ser un problema en cuanto a diferentes barras, se implementara una nueva variable para que el código sea genérico y responda de igual forma para cualquier tipo de barras. En el código debajo de la selección para la línea se debe colocar en comentario //SELECCION DE LA BARRA// //I.M.=1 //OTRAS=0 y apoyados de un int seguido de la barra = a la barra que se utilice se tendrá la variable para la barra.

```
//COLOR DE LA LINEA SEGUN LA BARRA DE SENSORES//
//I.M. BLANCO=1 NEGRO=0
//OTRAS BARRAS BLANCO=0 NEGRO=1
int linea=0;

///SELECCION DE LA BARRA//
//I.M=1
//OTRAS=0
int barra=0;
```

Imagen 242. Selección de barra.

En la parte de lectura se debe realizar un ajuste a la ponderación, se debe invertir dependiendo de la barra que se utilice. Se debe colocar una condición if que indique que si la barra es igual a 1 se debe realizar la suma ponderada con esa condición:

if(barra==1){sumap=(1500*digital[15]+1400*digital[14]+1300*digital[13]+1200*digital[12]+1100*digital[11]+1000*digital[10]+900*digital[9]+800*digital[8]+700*digital[7]+600*digital[6]+500*digital[5]+400*digital[4]+300*digital[3]+200*digital[2]+100*digital[1]+0*digital[0]);}

Esta misma línea de código se debe copiar y pegar justo debajo de la primera pero cambiando la condición de la barra if a 0 en vez de 1 indicando que la barra utilizada es otra distinta a la de Ingeniero Maker pero además también se debe cambiar la ponderación siendo 1500*digital[15] así hasta 0*digital[0]:

```
if(barra==0){sumap=(1500*digital[0]+1400*digital[1]+1300*digital[2]+1200*digital[3]+110
0*digital[4]+1000*digital[5]+900*digital[6]+800*digital[7]+700*digital[8]+600*digital[9]+500
*digital[10]+400*digital[11]+300*digital[12]+200*digital[13]+100*digital[14]+0*digital[15]);}
```

Con esto se finaliza el programa para la lectura y calibración de los sensores, garantizando que el programa funcione de forma correcta tanto para un fondo blanco como para un fondo negro, con una barra de sensores, como con otra.

Código completo:

```
#define s0        14
#define s1        15
#define s2        16
#define s3        17
#define AI        A4
#define LED_ON    12
#define LED       13
#define BOTON     9
int sensores [16];
```

```
int digital [16];

int lectura_fondo[16];

int lectura_linea[16];

int umbral[16];

//int umbral=600;

long int sumap, suma, pos, poslast, position;



//COLOR DE LA LINEA SEGUN LA BARRA DE SENSORES//

//I.M. BLANCO=1 NEGRO=0

//OTRAS BARRAS BLANCO=0 NEGRO=1

int linea=0;



///SELECCION DE LA BARRA//

//I.M=1

//OTRAS=0

int barra=0;
```

```
void setup() {
Serial.begin(115200);
pinMode(s0,OUTPUT);
pinMode(s1,OUTPUT);
pinMode(s2,OUTPUT);
pinMode(s3,OUTPUT);
pinMode(LED,OUTPUT);
pinMode(LED_ON,OUTPUT);
pinMode(BOTON,INPUT);
digitalWrite(LED_ON,1);
digitalWrite(LED,1);
while(digitalRead(BOTON));
for(int i=0;i<50;i++){
  fondos();
  digitalWrite(LED,0);
  delay(20);
  digitalWrite(LED,1);
  delay(20);
```

```
}
while(digitalRead(BOTON));
for(int i=0;i<50;i++){
  lineas();
  digitalWrite(LED,0);
  delay(20);
  digitalWrite(LED,1);
  delay(20);
}
promedio();
while(digitalRead(BOTON));
digitalWrite(LED,0);
}

void loop() {
  position=lectura();
Serial.println(position);
```

```
}

void fondos() {

  for(int i=0;i<16;i++){

  digitalWrite(s0,i&0x01);

  digitalWrite(s1,i&0x02);

  digitalWrite(s2,i&0x04);

  digitalWrite(s3,i&0x08);

  lectura_fondo[i]=analogRead(AI);

  Serial.print(lectura_fondo[i]);

  Serial.print("\t");

}

Serial.println(" ");



}

void lineas() {

   for(int i=0;i<16;i++){

  digitalWrite(s0,i&0x01);

  digitalWrite(s1,i&0x02);
```

```
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  lectura_linea[i]=analogRead(AI);
  Serial.print(lectura_linea[i]);
  Serial.print("\t");
}
Serial.println(" ");

}
void promedio(){
  for(int i=0;i<16;i++){
  umbral[i]=(lectura_fondo[i]+lectura_linea[i])/2;
  Serial.print(umbral[i]);
  Serial.print("\t");
}
Serial.println(" ");
}
```

```
    int lectura(void) {


/*digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[0]=analogR
ead(AI);

    digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[1]=analog
Read(AI);

    digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[2]=analog
Read(AI);

    digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[3]=analog
Read(AI);

    digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[4]=analog
Read(AI);

    digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[5]=analog
Read(AI);

    digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[6]=analog
Read(AI);

    digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[7]=analog
Read(AI);

    digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[8]=analog
Read(AI);
```

```
    digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[9]=analog
Read(AI);

    digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[10]=analog
Read(AI);

    digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[11]=analog
Read(AI);

    digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[12]=analog
Read(AI);

    digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[13]=analog
Read(AI);

    digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[14]=analog
Read(AI);

    digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[15]=analog
Read(AI);*/


    /*for(int i=0;i<16;i++){
      if(sensores[i]<=umbral){
        sensores[i]=0;
      }
      Serial.print(sensores[i]);
```

```
  Serial.print("\t");
 }
 Serial.println(" ");*/

 for(int i=0;i<16;i++){
  digitalWrite(s0,i&0x01);
  digitalWrite(s1,i&0x02);
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  sensores[i]=analogRead(AI);
  if(linea==0){if(sensores[i]<=umbral){digital[i]=0;}else{digital[i]=1;}}
  if(linea==1){if(sensores[i]<=umbral){digital[i]=1;}else{digital[i]=0;}}
  Serial.print(digital[i]);
  Serial.print("\t");
 }
 //Serial.println(" ");
 if(barra==1){sumap=(1500*digital[15]+1400*digital[14]+1300*digital[13]+1200*digital[12]
+1100*digital[11]+1000*digital[10]+900*digital[9]+800*digital[8]+700*digital[7]+600*digital[
6]+500*digital[5]+400*digital[4]+300*digital[3]+200*digital[2]+100*digital[1]+0*digital[0]);}
```

```
    if(barra==0){sumap=(1500*digital[0]+1400*digital[1]+1300*digital[2]+1200*digital[3]+110
0*digital[4]+1000*digital[5]+900*digital[6]+800*digital[7]+700*digital[8]+600*digital[9]+500
*digital[10]+400*digital[11]+300*digital[12]+200*digital[13]+100*digital[14]+0*digital[15]);}

    suma=(digital[0]+digital[1]+digital[2]+digital[3]+digital[4]+digital[5]+digital[6]+digital[7]+d
igital[8]+digital[9]+digital[10]+digital[11]+digital[12]+digital[13]+digital[14]+digital[15]);

    pos=sumap/suma;

    if(poslast<=100 && pos==-1){

      pos=0;

    }

    if(poslast>=1400 && pos==-1){

      pos=1500;

    }

    poslast=pos;

    return pos;

    }
```

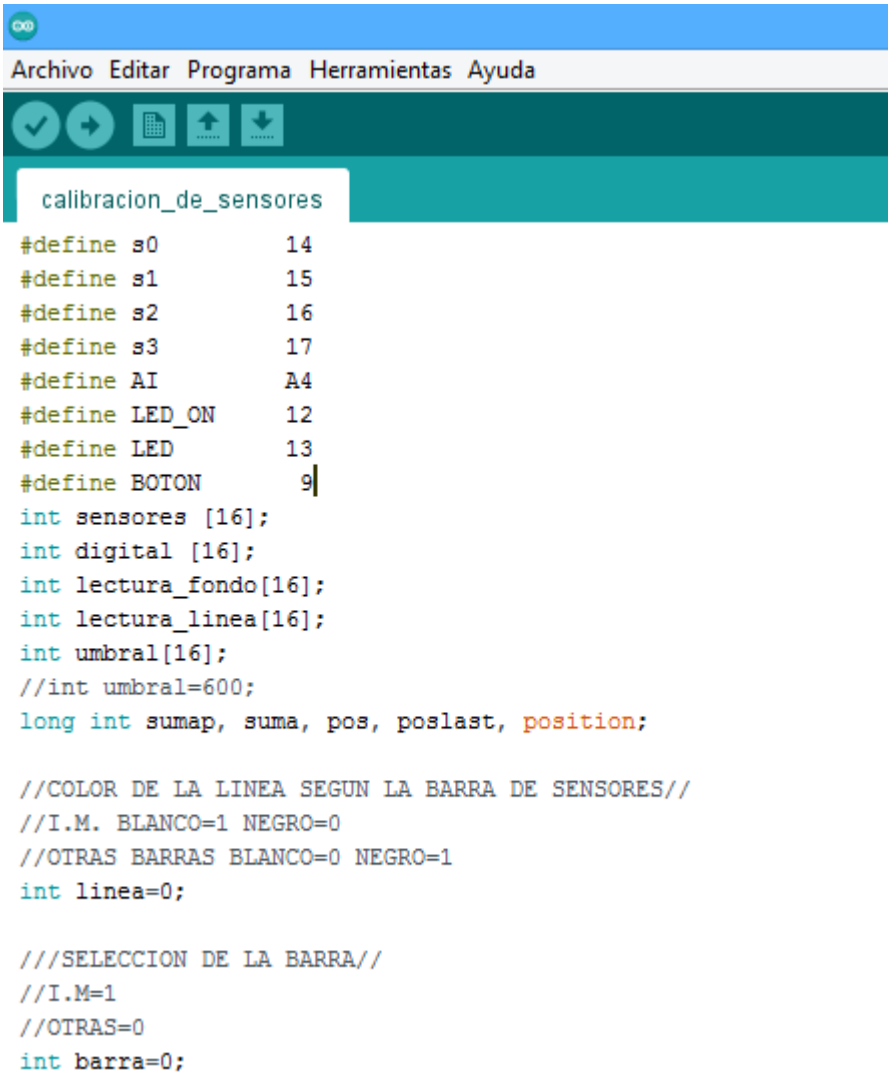


Imagen 243. Código para la calibración y lectura de sensores 1.

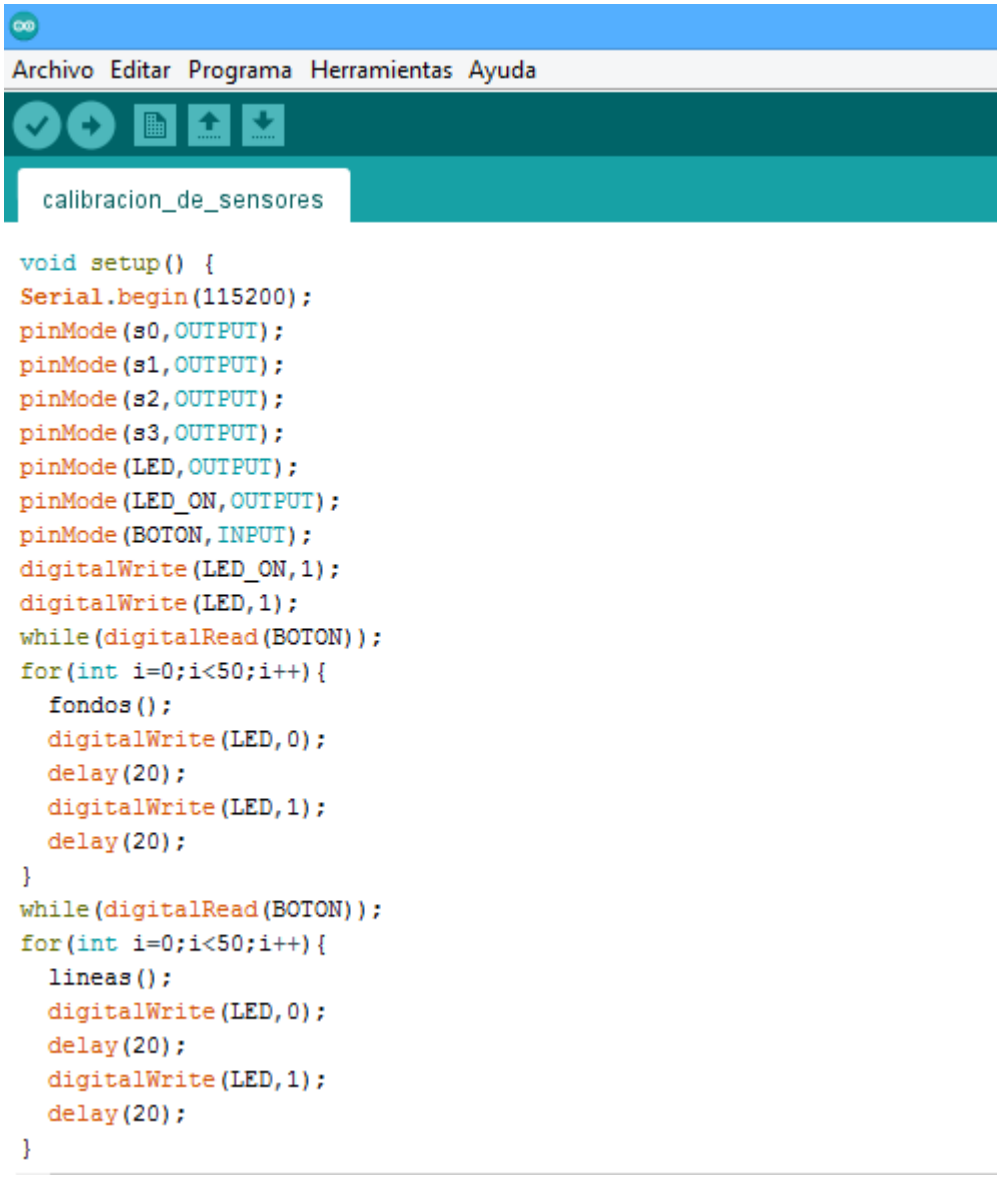


Imagen 244. Código para la calibración y lectura de sensores 2.

Imagen 245. Código para la calibración y lectura de sensores 3.

Imagen 246. Código para la calibración y lectura de sensores 4.

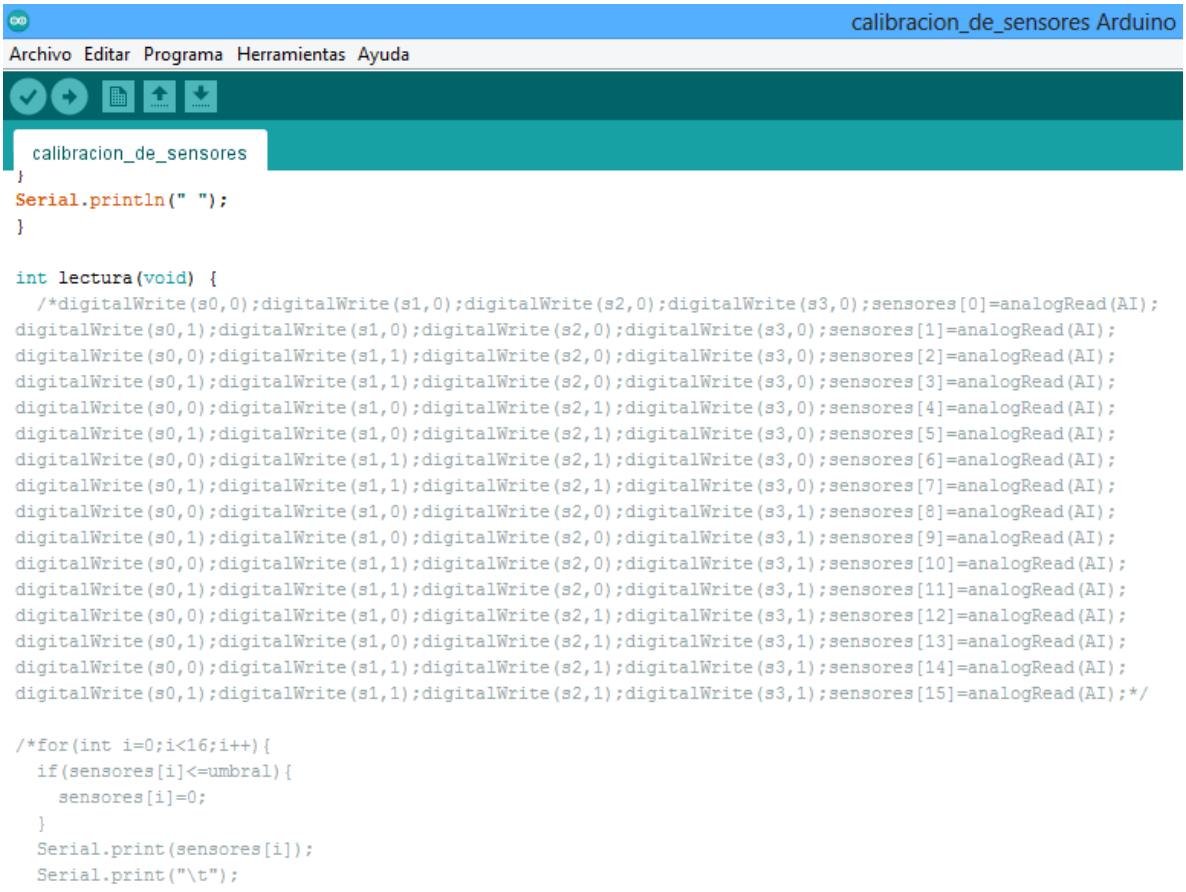


Imagen 247. Código para la calibración y lectura de sensores 5.

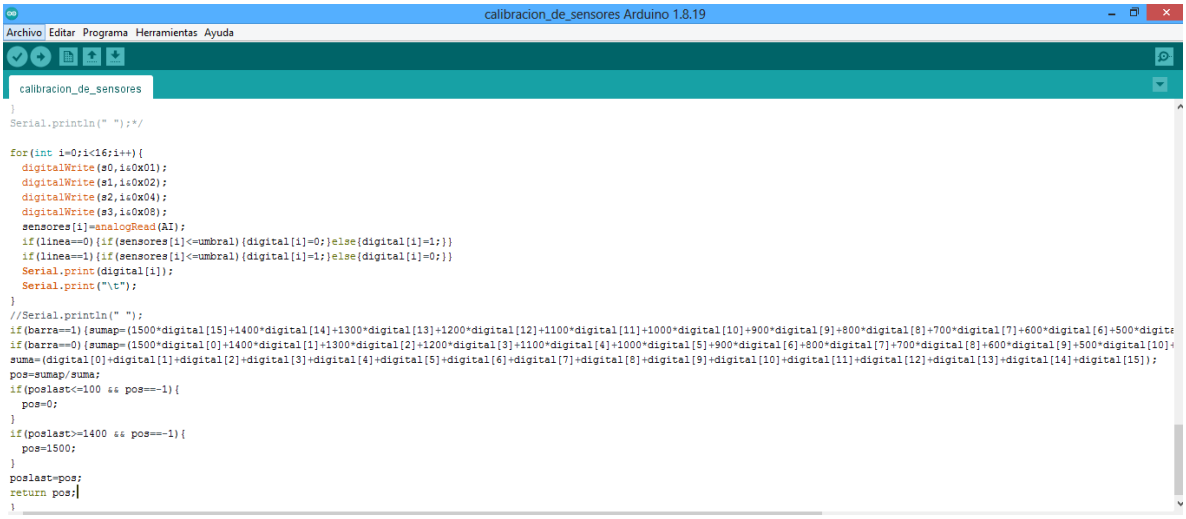


Imagen 248. Código para la calibración y lectura de sensores 6.

5.7 Implementación de pid al programa para el seguidor de línea

Para realizar la implementación del PID al código se tomó como base el modelo de PID del seguidor de linea “Lamborghino”, siendo un seguidor de línea muy popular y del cual muchas personas han tomado como referencia para el desarrollo de seguidores de línea y su implementación dentro Arduino para el control con el algoritmo de corrección de error que se implementara para el control del seguidor de línea. Además también se utilizara una librería ESC.h que se debe descargar de la página Butduino.com que será de utilidad para la función de la turbina. Iniciando primeramente con la aplicación de esta librería, se tiene que descargar de la página. Esta librería servirá para el control de las turbinas EDF30, la cual es utilizada para el proyecto, pero también para la EDF27, QXMotors, etc. Teniendo un código genérico que permita a partir de esta librería el control de varios tipos de turbinas. Lo primero a realizar es abrir el programa para la lectura y calibración de la barra de sensores, posteriormente la librería descargada debe ser descomprimida e incluida desde el programa de Arduino, para esto se debe ir a la pestaña Programa>Incluir Librería>Añadir Biblioteca ZIP

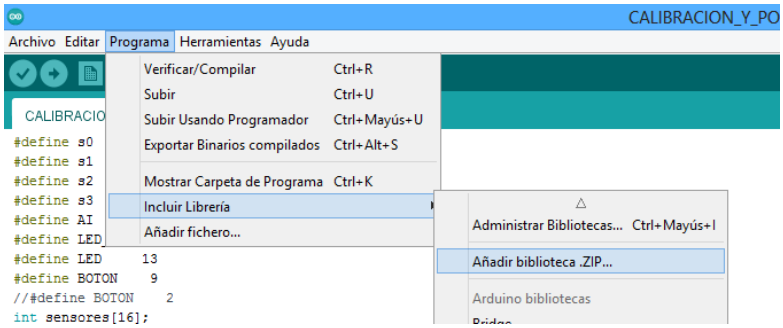


Imagen 249. Añadir librería ESC.h

Una vez seleccionada la opción Añadir Biblioteca ZIP se abrirá un buscador, se selecciona la carpeta que contenga la librería y listo, la librería estará cargada en el programa.

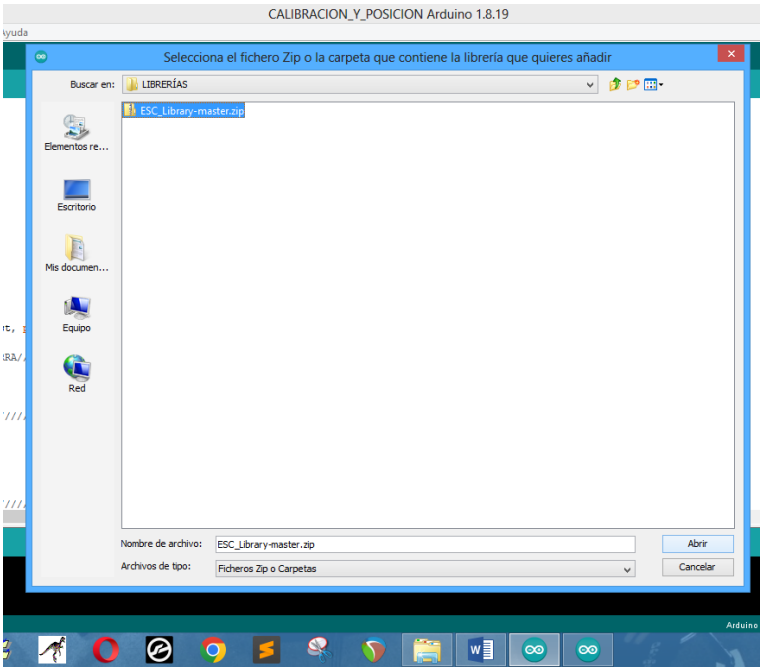


Imagen 250. Búsqueda del archivo Librería ESC.h.

Cargada la librería en la parte inferior del programa se mostrara un mensaje de aviso que indicara que la librería ha sido agregada correctamente. La librería debe ser cargada al principio del programa con la siguiente línea de código: #include <ESC.h> la segunda línea de código será utilizada para definir en qué pin se encuentra conectado el ESC, para el esquema realizado el pin utilizado para la conexión fue el pin D10 (pin 10), para definir este pin en base al ESC se debe usar la siguiente línea de código: #define ESC_PIN 10. Siguiente a esto se debe otorgar un nombre al ESC, y para designarle un nombre se debe llamar a la librería en cuestión desde el código, entonces para este caso se debe agregar una línea de código iniciando por el ESC seguido de la turbina utilizada, en este caso la EDF30, seguido de un paréntesis con ESC_PIN, el pin utilizado de la siguiente manera: ESC EDF30(ESC_PIN); esta línea de código será colocada debajo de la línea que define al botón en el pin 2.

```
#include <ESC.h>
#define ESC_PIN    10
#define s0         14
#define s1         15
#define s2         16
#define s3         17
#define AI         A4
#define LED_ON     12
#define LED        13
#define BOTON       2


ESC EDF30(ESC_PIN);
int sensores [16];
```

Imagen 251. Líneas de código para el ESC.h.

Como función extra de la librería agregada, en los ejemplos de programación que tiene Arduino se ha agregado el ejemplo de cómo funciona la turbina y sus líneas de código, se puede ver cuál es el mínimo para apagar la turbina y el máximo para encenderla, velocidad mínima y máxima de la turbina, y la explicación del programa en comentarios. De este programa de ejemplo se puede extraer una línea de código que funciona para iniciar el proceso de la turbina, el TIMER 1 en este caso. Para esto en el void setup entre la declaración de la salida de pines y los LEDs se debe copiar o escribir la siguiente línea de código: EDF30.init(); de esta manera se estará dando un retraso, un delay de unos 500 milisegundos para que la turbina pueda iniciar su proceso de forma correcta, y el Serial.begin se debe dejar como comentario y colocar al principio del void setup.

```
pinMode(BOTON,INPUT);
EDF30.init();
digitalWrite(LED_ON,1);
```

Imagen 252. Línea de código para inicio de la turbina.

Para esta parte del programa se utiliza prácticamente todo en cuanto a programación se ha realizado, porque este es el programa final, por esto es necesario abrir el programa de los motores.

Para iniciar se debe copiar toda la definición de los pines para los motores, y pegarlos con las demás definiciones del programa de lectura y calibración

```
#include <ESC.h>
#define ESC_PIN    10
#define s0         14
#define s1         15
#define s2         16
#define s3         17
#define AI         A4
#define LED_ON     12
#define LED        13
#define BOTON       2
#define  pwmi       3 //PWMA MOTOR IZQUIERDO
#define  izq1       5
#define  izq2       4
#define  pwmd      11 //PWMB MOTOR DERECHO
#define  der1       6
#define  der2       7
```

Imagen 253. Unión de programa motores.

Ahora se debe copiar todo lo que se encuentra dentro del void setup para los motores, tanto la declaración de pines como la configuración del TIMER2 y se debe pegar debajo del Serial.begin que quedo como comentario, por último se debe copiar la función de los motores, void motores y pegar al final de todo el código. Y para este punto deben dejar en comentarios todos los Serial que contenga el código para que la programación no se vea afectada debido a que no se necesitan hacer lecturas.

Lo siguiente es declarar un pin que encienda cuando el modulo arrancador recibe una señal, este pin puede ser nombrado como “GO” que responde al pin 2, la línea de código deberá ser colocada debajo de la declaración de la variable BOTON con el pin 9, con el siguiente código: #define GO 2 y para que esta línea de código funcione, debe ser declarada como entrada junto con los demás

pines justo encima de la línea que inicia el proceso para la turbina con el siguiente código: pinMode(GO,INPUT);

```
pinMode(BOTON,INPUT);
pinMode(GO,INPUT);
EDF30.init();
digitalWrite(LED_ON,1);
```

Imagen 254. Declarando GO como entrada.

Para poner al seguidor en marcha se sabe que primero se debe presione el pulsador una vez para que comience la calibración en fondo, una segunda vez para que lo haga en la línea, y la tercera era para las lecturas de posición o el promedio, en este caso se cambiará la última pulsación de "BOTON" por la función "GO" para que el seguidor se quede en espera recibiendo un 0 lógico mientras el operador por medio del control no indique el arranque del seguidor, entonces al realizar esta acción el operador por medio del control enviara un 1 lógico para activar al seguidor, por tanto en el programa, para el arreglo promedio será reemplazado "BOTON" por "GO" y será negada esta función para que sea 0

```
promedio();
while(!digitalRead(GO));
digitalWrite(LED,0);
}
```

Imagen 255. Cambiando variables.

5.7.1 Función PID

Para realizar las funciones de PID es necesario en un principio crear nuevas funciones, pero también es importante poder distinguir claramente donde inicia el PID por tanto como comentario se recomienda colocar //////////PID//////// para indicar que inicia la parte de PID justo por encima de void setup, seguido de este comentario se deben declarar 3 variables de tipo flotante, una para

la parte proporcional, para la parte derivativa y para la parte integral, KP=0 para la proporcional, KD para la derivativa y KI para la integral, todas con valor 0, después a la variable int se agregaran variables para velocidad, velocidad de la turbina, velocidad del freno hacia adelante, velocidad del freno hacia atrás, velocidad de prueba de la turbina, tal como lo muestra la siguiente línea de código:

```
/////////////// PID//////////////////
float KP=0;//constante proporcional
float KD=0;//constante derivativa
float KI=0;//constante integral
int vel=180;//VELOCIDAD MÁXIMA DEL ROBOT MÁXIMA 255
int turbina=1800;//VELOCIDAD DE LA TURBINA MINIMA 1000 MÁXIMA2000
int veladelante=200;//VELOCIDAD DEL FRENO DIRECCIÓN ADELANTE
int velatras=100;//VELOCIDAD DEL FRENO DIRECCIÓN ATRÁS
int veltest=1500;//VELOCIDAD DE TEST DE LA TURBINA
```

Hasta este punto, las variables que se han colocado se entienden de manera fácil y simple, los comentarios indican que función tiene cada variable. La parte en cuanto a variables presenta dos variables para turbina, la primera es para la velocidad de la turbina en función al seguidor, cuando el seguidor está corriendo sobre pista, la segunda función de la turbina es la velocidad de la turbina, pero con el seguidor aún en estado de espera, o sea cuando aún no está corriendo, esto se hace con el fin de prolongar la vida útil de la turbina evitando que la turbina inicie con una velocidad alta

casi en su máxima potencia desde la fase de posición para el seguidor, entonces la primera velocidad es menor, con el fin de evitar desgaste innecesario con la velocidad máxima.

5.7.2 Variable Integral

Para la variable integral se deben introducir 6 errores, todos a partir dela variable int, comenzando por 1

error1, error 2, etc. Con un valor igual a 0

```
/// DATOS PARA LA INTEGRAL//////

int error1=0;

int error2=0;

int error3=0;

int error4=0;

int error5=0;

int error6=0;
```

5.7.3 Variable PID

Por último se deben declarar las variables propias del PID, que corresponden a la parte proporcional, integral y derivativa, pero también las variables para diferencial encargada del control de la corrección de error, todas con el valor a 0, la variable setpoint por otro lado deberá tener el valor de 750, ¿porque 750? Porque es el punto medio de la barra de sensores, que se vio en la calibración, justo en la parte de calibración se menciona que 750 es el setpoint para el seguidor de línea, y este es un valor que teóricamente es correcto pero siempre pueden existir variaciones,

tanto por la electrónica de la barra, como la barra misma. En caso de utilizar un setpoint mayor o menor, el seguidor no tendrá la misma respuesta debido a que el setpoint no estará ubicado al centro de la barra de sensores. Por tanto 750 es el número ganador para el proyecto.

```
//////////VARIABLE PID///////////////

int proporcional=0;

int integral=0;

int derivativo=0;

int diferencial=0;

int last_prop;

int setpoint=750;
```

Una vez declaradas las variables de PID se debe crear la función para el PID, que se puede ubicar entre las líneas de código para la calibración de los sensores y entre las líneas de código para los motores. Entonces se inicia con la función void PID(){ seguido de la variable proporcional que será igual a la posición menos el setpoint, para que se realice la corrección de la posición respecto al setpoint. La ecuación para la variable derivativa debe ser igual a la variable proporcional menos una proporcional anterior. Seguido de la derivativa se continua con la variable integral siendo igual a la suma de sus 6 errores, una vez colocados los 6 errores se debe colocar nuevamente la última proporcional igual a la proporcional para que luego de repetirse el PID tenga en cuenta el valor anterior que tenía esta variable y después se tienen que colocar todos los errores igualados uno al siguiente, en este caso error6=error5 y así para todos hasta el error 1=proporcional con el fin de que la integral se vaya acumulando. Después se debe introducir la variable diferencial, que tiene

que calcular el PID como tal, teniendo int diferencial siendo igual a la proporcional*KP+derivativo*KD+integral*KI. Teniendo la variable diferencial se debe limitar la diferencial para que no pase de la velocidad máxima, para esto se utiliza una condicional if que indique que si la diferencial es mayor que la velocidad la diferencial debe ser igual a la velocidad, de esta manera existe un límite y no se puede pasar de ahí. También se debe colocar su caso contrario, con un else if donde la diferencial es menor a menos la velocidad (menor porque el valor varía) la diferencia será igual a menos la velocidad, como resultado se está limitando a la variable velocidad tanto positiva como negativamente para que los motores vayan hacia adelante o hacia atrás. Lo siguiente es condicionar el movimiento de los motores según la posición teniendo a la diferencial siendo menor a 0 y que primero el motor izquierdo vaya a la velocidad máxima y el motor derecho regule su velocidad en base a la posición,(motores vel, para el motor izquierdo más vel+ diferencial) seguido de esto se hace lo contrario para el motor derecho pero la velocidad para el izquierdo será menos la diferencia, y la velocidad máxima será ahora para el derecho (motores vel-diferencial, vel). Con esto el PID estaría completo, lo recomendable seria compilar el programa para saber si existe algún error.

```
void PID(){
  proporcional=pos-setpoint;
  derivativo=proporcional-last_prop;
  integral=error1+error2+error3+error4+error5+error6;
  last_prop=proporcional;
  error6=error5;
```

```
    error5=error4;
    error4=error3;
    error3=error2;
    error2=error1;
    error1=proporcional;
    int diferencial=(proporcional*KP) + (derivativo*KD) + (integral*KI);
    if(diferencial > vel) diferencial=vel;
    else if(diferencial < -vel) diferencial=-vel;
    (diferencial < 0)?
    motores(vel, vel+diferencial):motores(vel-diferencial, vel);
  }
```

5.8 Frenos

Para el desarrollo de los frenos, que en realidad es el descenso y giro contrario del motor para la reducción de la velocidad del seguidor de línea será necesario el desarrollo de un nuevo arreglo denominado “frenos” justo por debajo de las líneas de código para el PID, entonces con ayuda de un void se realizará la siguiente línea de código: void frenos(){ para dar inicio a la parte de frenos, con un condicional if, si la posición es menor o igual a 250 o sea que si el seguidor de línea se sale por la parte derecha, el motor izquierdo deberá girar hacia atrás, mientras el motor derecho continua con su sentido de giro, para que el seguidor corrija su dirección y vuelva a la línea, después se tiene que definir que los motores tengan las variables veladelante, -velatras que fueron colocados para

el PID, después se repetirá la misma condicional if, siendo la posición mayor o igual a 1350, y para los motores será el proceso inverso, se colocara primero la velocidad negativa hacia atrás, y velocidad hacia adelante, con el siguiente código:

```
void frenos(){
 if(pos<=250){
   motores(veladelante, -velatras);
 }
 if(pos>=1350){
   motores(-velatras, veladelante);
 }
}
```

Habiendo definido la parte de los frenos es necesario llamar a todas estas funciones del arreglo dentro del void loop, entonces debajo del void loop se eliminara la posición=lectura, junto con el serial en comentario y se colocará con ayuda de una variable int un go que leerá todo lo que se encuentre en el pin 2 (GO), además se debe colocar un bucle infinito while, en el cual si es que se presiona el botón (se envía un 1 lógico) del módulo de arranque, éste entrará al bucle infinito en el que funcionará directamente, por lo cual será necesario volver a llamar a la variable GO para cuando se apague el módulo arrancador se salga del bucle infinito. Seguido del proceso para la variable GO se deberá llamar a la variable frenos, también a la lectura de sensores, y al PID, en este caso se debe condicionar con un if, si go es igual a 0, o sea que se haya apagado al módulo arrancador desde el control, se romperá el bucle, no sin antes haber detenido un poco a los motores

con valores de -20 y posteriormente con ayuda de un break se romperá el bucle infinito, para después ingresar a un nuevo bucle infinito con el cual los motores estarán apagados, de esta manera se estará indicando que aquí se termina el programa, o sea cuando se apague el módulo arrancador mientras el seguidor este en pista funcionando se detendrá de forma inmediata y luego de salir del bucle infinito el seguidor se apagará y se estará deteniendo el programa. El código que representa esta explicación es el siguiente:

```
int go=digitalRead(GO);
while(true){
  int go=digitalRead(GO);
  frenos();
  lectura();
  PID();
  if(go==0){
    motores(-20,-20);
    break;
  }
}
while(true){
  motores(0,0);
```

```
 EDF30.setSpeed(1000);
```

5.9 Activación de la turbina

En un principio se agregó la librería para la turbina y el código para iniciar la turbina pero no la activación, y la activación no se colocó al momento de haberla iniciado debido a que faltaba colocar valores como la velocidad final de la turbina y la velocidad de prueba que se incluyeron el PID. Por tanto en esta parte se realiza la activación de la turbina. Para esto debajo de la parte de promedio se deberá colocar la instrucción while(digitalRead(BOTON)); para encender a la turbina, para esto se utiliza un bucle for que inicie en 1000 y menor igual a 2000 sumándose de 1 en 1, después se coloca el nombre de la turbina yendo a una velocidad i con un delay de 2 milisegundos, para saber en qué tiempo se enciende la turbina, después de salir del bucle la turbina deberá esperar funcionando a la velocidad de espera designada en veltest, hasta que se presione el BOTON para después alcanzar la velocidad máxima designada para la turbina y por último con el control se activa el GO para que el seguidor corra en pista.

```
while(digitalRead(BOTON));
for(int i=1000;i<=2000;i++){
 EDF30.setSpeed(i);
 delay(2);
}
EDF30.setSpeed(veltest);
while(digitalRead(BOTON));
```

```
EDF30.setSpeed(turbina);

while(!digitalRead(GO));

digitalWrite(LED,0);

}
```

Con esto se ha concluido el programa, lo único que resta por hacer es compilar el programa para verificar que tanto la turbina, la calibración, lectura de los sensores y los motores funcionen correctamente, no obstante aún no se podrían realizar pruebas de recorrido hasta este momento, debido a que aún resta asignar valores al PID para que el seguidor pueda correr en pista.

5.9.1 Valores para el PID

El PID (control proporcional, integral y derivativo) es un mecanismo de control por realimentación que calcula la desviación o error entre un valor medido y el valor que se quiere obtener (set point, target position o punto de consigna), para aplicar una acción correctora que ajuste el proceso. En el caso del robot velocista, el controlador PID, (que es una rutina basada matemáticamente), procesara los datos del sensor, y lo utiliza para controlar la dirección (velocidad de cada motor), para de esta forma mantenerlo en curso.

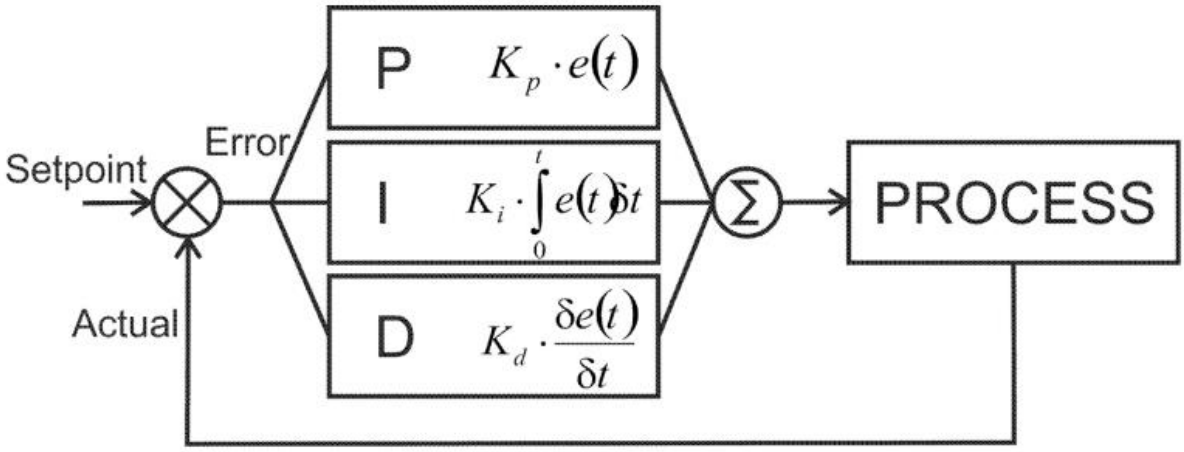


Diagrama de control del PID.

$$u(t) = k\left[e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{de(t)}{dt}\right] = P + I + D$$

- Error: Se conoce como la diferencia entre la posición objetivo y la posición medida del error. (Que tan lejos del punto de consigna se encuentra el sensor, en nuestro caso el objetivo es tener los sensores centrados).

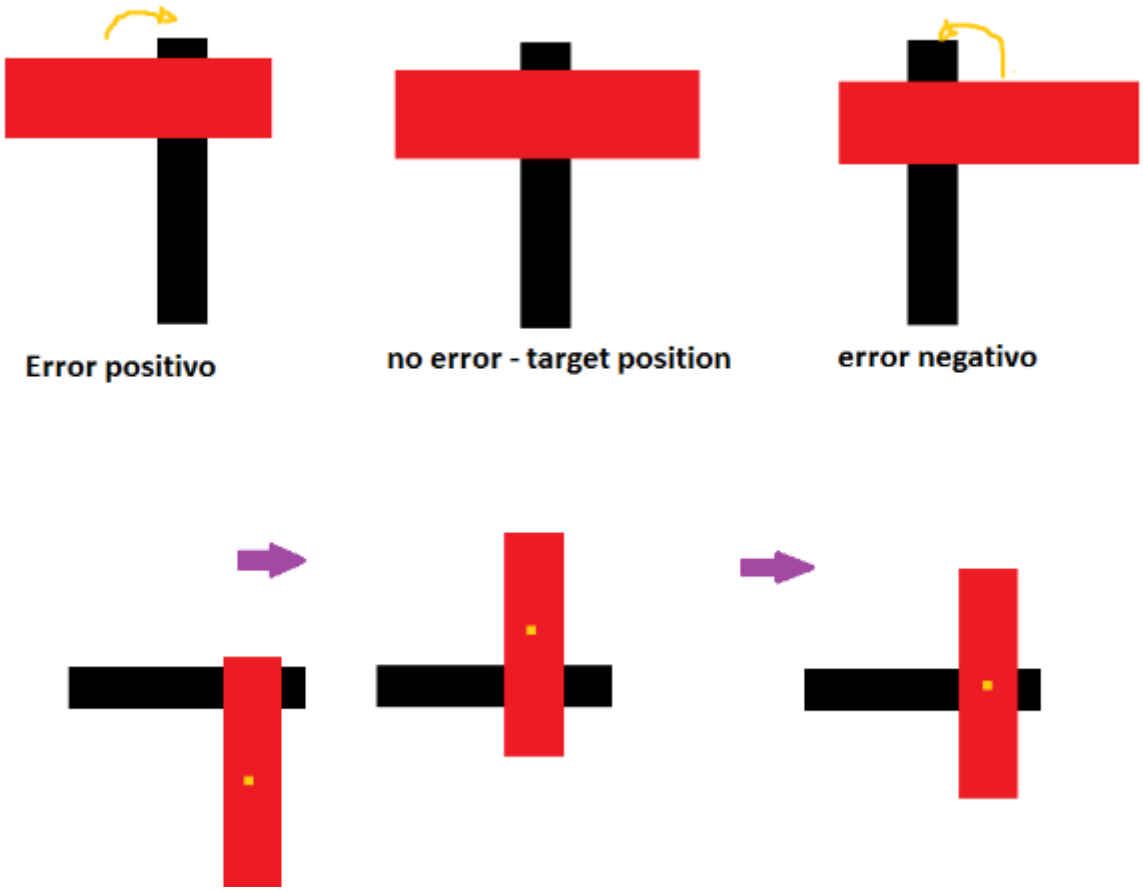


Imagen 256. Tipos de error.

-Set point o Target Position: Es cuando el error es 0 (cero). En el caso del robot velocista es 750, la idea es siempre mantenerlo en la línea, o lo que es el caso de los sensores, mantenerlo centrado y así no se llegue a salir de la línea.

PARAMETROS: -Proporcional: Es la respuesta al error que se tiene que entregar de manera inmediata, es decir, si nos encontramos en el centro de la línea, los motores, tendrán en respuesta una velocidad de igual valor, si nos alejamos del centro, uno de los motores reducirá su velocidad y el otro aumentara. Proporcional=(posición) -punto consigna -Integral: La integral es la sumatoria de los errores acumulados, tiene como propósito el disminuir y eliminar el error en estado estacionario provocado por el modo proporcional, en otras palabras, si el robot velocista se

encuentra mucho tiempo alejado del centro (ocurre muchas veces cuando se encuentra en curvas), la acción integral se ira acumulando e ira disminuyendo el error hasta llegar al punto de consigna, Integral=Integral + proporcional pasado -Derivativo: Es la derivada del error, su función es mantener el error al mínimo, corrigiéndolo proporcionalmente con la mismo velocidad que se produce, de esta manera evita que el error se incremente, en otra palabra, anticipara la acción evitando así las oscilaciones excesivas. Derivativo=proporcional-proporcional pasado ->**CONSTANTES**: Factor (Kp) - Es un valor constante utilizado para aumentar o reducir el impacto de Proporcional. Si el valor es excesivo, el robot tendera responder inestablemente, oscilando excesivamente. Si el valor es muy pequeño, el robot responderá muy lentamente, tendiendo a salirse de las curvas Factor (Ki) - Es un valor constante utilizado para aumentar o reducir el impacto de la Integral, El valor excesivo de este provocara oscilaciones excesivas, Un valor demasiado bajo no causara impacto alguno. Factor (Kd) - Es un valor constante utilizado para aumentar o reducir el impacto de la Derivada. Un valor excesivo provocara una sobre amortiguación. Provocando inestabilidad. Salida pwm = (proporcional * Kp) + (derivativo * Kd) + (integral*Ki); ->**SINTONIZACION PID** Aquí viene el reto, la sintonización pid, es aquí donde se tendrá que buscar las constantes que correspondan a las características físicas del robot, la forma más fácil de hacerlo es por ensayo y error, hasta obtener el valor deseado. Aquí hay unos pasos que ayudarán mucho a buscar esas constantes: 1. Comience con Kp, Ki y Kd igualando 0 y trabajar con Kp primero. Pruebe establecer Kp a un valor de 1 y observar el robot. El objetivo es conseguir que el robot siga la línea, incluso si es muy inestable. Si el robot llega más allá y pierde la línea, reducir el valor de Kp. Si el robot no puede navegar por una vez, o parece lenta, aumente el valor Kp. 2. Una vez que el robot es capaz de seguir un poco la línea, asignar un valor de 1 a Kd .Intente aumentar este valor hasta que vea menos oscilaciones. 3. Una vez que el robot es bastante estable

en la línea siguiente, asigne un valor de 0,5 a 1,0 a Ki. Si el valor de Ki es demasiado alta, el robot se sacudirá izquierda y derecha rápidamente. Si es demasiado baja, no se verá ninguna diferencia perceptible. El Integral es acumulativo por lo tanto el valor Ki tiene un impacto significativo. Puede terminar ajustando por 0,01 incrementos. 4. Una vez que el robot está siguiendo la línea con una buena precisión, se puede aumentar la velocidad y ver si todavía es capaz de seguir la línea. La velocidad afecta el controlador PID y requerirá resintonizar como los cambios de velocidad.

Los valores para el PID en base a la información anterior y la información recabada para el proyecto en pista por prueba y error corresponden a los siguientes:

```
////////////////PID////////////////////

float KP=.5;//constante proporcional

float KD=7;//constante derivativa

float KI=0.0052;//constante integral

int vel=180;//VELOCIDAD MÁXIMA DEL ROBOT MÁXIMA 255

int turbina=1800;//VELOCIDAD DE LA TURBINA MINIMA 1000 MÁXIMA2000

int veladelante=200;//VELOCIDAD DEL FRENO DIRECCIÓN ADELANTE

int velatras=100;//VELOCIDAD DEL FRENO DIRECCIÓN ATRÁS

int veltest=1500;//VELOCIDAD DE TEST DE LA TURBINA
```

5.10 Programa final

```
#include <ESC.h>

#define ESC_PIN    10

#define s0         14

#define s1         15

#define s2         16

#define s3         17

#define AI         A4

#define LED_ON     12

#define LED        13

#define BOTON      9

#define GO         2

#define pwmi       3 //PWMA MOTOR IZQUIERDO

#define izq1       5

#define izq2       4

#define pwmd       11 //PWMB MOTOR DERECHO

#define der1       6

#define der2       7
```

```
ESC EDF30(ESC_PIN);

int sensores [16];

int digital [16];

int lectura_fondo[16];

int lectura_linea[16];

int umbral[16];

//int umbral=600;

long int sumap, suma, pos, poslast, position;


//COLOR DE LA LINEA SEGUN LA BARRA DE SENSORES//

//I.M. BLANCO=1 NEGRO=0

//OTRAS BARRAS BLANCO=0 NEGRO=1

int linea=0;


///SELECCION DE LA BARRA//

//I.M=1
```

```
//OTRAS=0

int barra=0;


////////////////PID////////////////////

float KP=.5;//constante proporcional

float KD=7;//constante derivativa

float KI=0.0052;//constante integral

int vel=180;//VELOCIDAD MÁXIMA DEL ROBOT MÁXIMA 255

int turbina=1800;//VELOCIDAD DE LA TURBINA MINIMA 1000 MÁXIMA2000

int veladelante=200;//VELOCIDAD DEL FRENO DIRECCIÓN ADELANTE

int velatras=100;//VELOCIDAD DEL FRENO DIRECCIÓN ATRÁS

int veltest=1500;//VELOCIDAD DE TEST DE LA TURBINA


/////DATOS DE LA VARIABLE INTEGRAL///////

int error1=0;

int error2=0;

int error3=0;

int error4=0;
```

```
int error5=0;

int error6=0;

//////////////////////////


//////////VARIABLE PID///////////////

int proporcional=0;

int integral=0;

int derivativo=0;

int diferencial=0;

int last_prop;

int setpoint=750;




void setup() {

//Serial.begin(115200);

//TCCR2B = TCCR2B & B11111000 | B00000001;

//set timer 2 divisor to     1 for PWM frequency of 31372.55 Hz
```

TCCR2B = TCCR2B & B11111000 | B00000010;

//set timer 2 divisor to 8 for PWM frequency of 3921.16 Hz

//TCCR2B = TCCR2B & B11111000 | B00000011;

//set timer 2 divisor to 32 for PWM frequency of 980.39 Hz

//TCCR2B = TCCR2B & B11111000 | B00000100;

//set timer 2 divisor to 64 for PWM frequency of 490.20 Hz (The DEFAULT)

//TCCR2B = TCCR2B & B11111000 | B00000101;

//set timer 2 divisor to 128 for PWM frequency of 245.10 Hz

//TCCR2B = TCCR2B & B11111000 | B00000110;

//set timer 2 divisor to 256 for PWM frequency of 122.55 Hz

//TCCR2B = TCCR2B & B11111000 | B00000111;

//set timer 2 divisor to 1024 for PWM frequency of 30.64 Hz

```
pinMode (izq1, OUTPUT);

pinMode (izq2, OUTPUT);

pinMode (der1, OUTPUT);

pinMode (der2, OUTPUT);

pinMode(s0,OUTPUT);

pinMode(s1,OUTPUT);

pinMode(s2,OUTPUT);

pinMode(s3,OUTPUT);

pinMode(LED,OUTPUT);

pinMode(LED_ON,OUTPUT);

pinMode(BOTON,INPUT);

pinMode(GO,INPUT);

EDF30.init();

digitalWrite(LED_ON,1);

digitalWrite(LED,1);

while(digitalRead(BOTON));

for(int i=0;i<50;i++){

  fondos();
```

```
  digitalWrite(LED,0);
  delay(20);
  digitalWrite(LED,1);
  delay(20);
}
while(digitalRead(BOTON));
for(int i=0;i<50;i++){
  lineas();
  digitalWrite(LED,0);
  delay(20);
  digitalWrite(LED,1);
  delay(20);
}
promedio();
while(digitalRead(BOTON));
for(int i=1000;i<=2000;i++){
  EDF30.setSpeed(i);
  delay(2);
```

```
}
EDF30.setSpeed(veltest);
while(digitalRead(BOTON));
EDF30.setSpeed(turbina);
while(!digitalRead(GO));
digitalWrite(LED,0);
}

void loop() {
int go=digitalRead(GO);
while(true){
  int go=digitalRead(GO);
  frenos();
  lectura();
  PID();
  if(go==0){
    motores(-20,-20);
    break;
```

```
  }
}
while(true){
  motores(0,0);
  EDF30.setSpeed(1000);

}
}

void fondos() {
  for(int i=0;i<16;i++){
  digitalWrite(s0,i&0x01);
  digitalWrite(s1,i&0x02);
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  lectura_fondo[i]=analogRead(AI);
  //Serial.print(lectura_fondo[i]);
  //Serial.print("\t");
```

```
}
//Serial.println(" ");


}
void lineas() {
   for(int i=0;i<16;i++){
  digitalWrite(s0,i&0x01);
  digitalWrite(s1,i&0x02);
  digitalWrite(s2,i&0x04);
  digitalWrite(s3,i&0x08);
  lectura_linea[i]=analogRead(AI);
  //Serial.print(lectura_linea[i]);
  //Serial.print("\t");
}
//Serial.println(" ");


}
void promedio(){
```

```
    for(int i=0;i<16;i++){
    umbral[i]=(lectura_fondo[i]+lectura_linea[i])/2;
    //Serial.print(umbral[i]);
    //Serial.print("\t");
   }
   //Serial.println(" ");
   }

   int lectura(void) {

/*digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[0]=ana
logRead(AI);
   digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,0);sensores[1]=an
alogRead(AI);
   digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[2]=an
alogRead(AI);
   digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,0);sensores[3]=an
alogRead(AI);
   digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[4]=an
alogRead(AI);
```

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,0);sensores[5]=an alogRead(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[6]=an alogRead(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,0);sensores[7]=an alogRead(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[8]=an alogRead(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,0);digitalWrite(s3,1);sensores[9]=an alogRead(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[10]=a nalogRead(AI);

digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,0);digitalWrite(s3,1);sensores[11]=a nalogRead(AI);

digitalWrite(s0,0);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[12]=a nalogRead(AI);

digitalWrite(s0,1);digitalWrite(s1,0);digitalWrite(s2,1);digitalWrite(s3,1);sensores[13]=a nalogRead(AI);

digitalWrite(s0,0);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[14]=a nalogRead(AI);

```
    digitalWrite(s0,1);digitalWrite(s1,1);digitalWrite(s2,1);digitalWrite(s3,1);sensores[15]=a
nalogRead(AI);*/


    /*for(int i=0;i<16;i++){
      if(sensores[i]<=umbral){
        sensores[i]=0;
      }
      Serial.print(sensores[i]);
      Serial.print("\t");
    }
    Serial.println(" ");*/


    for(int i=0;i<16;i++){
      digitalWrite(s0,i&0x01);
      digitalWrite(s1,i&0x02);
      digitalWrite(s2,i&0x04);
      digitalWrite(s3,i&0x08);
      sensores[i]=analogRead(AI);
```

```
    if(linea==0){if(sensores[i]<=umbral){digital[i]=0;}else{digital[i]=1;}}
    if(linea==1){if(sensores[i]<=umbral){digital[i]=1;}else{digital[i]=0;}}
    //Serial.print(digital[i]);
    //Serial.print("\t");
  }
  //Serial.println(" ");
  if(barra==1){sumap=(1500*digital[15]+1400*digital[14]+1300*digital[13]+1200*digital[12]+1100*digital[11]+1000*digital[10]+900*digital[9]+800*digital[8]+700*digital[7]+600*digital[6]+500*digital[5]+400*digital[4]+300*digital[3]+200*digital[2]+100*digital[1]+0*digital[0]);}
  if(barra==0){sumap=(1500*digital[0]+1400*digital[1]+1300*digital[2]+1200*digital[3]+1100*digital[4]+1000*digital[5]+900*digital[6]+800*digital[7]+700*digital[8]+600*digital[9]+500*digital[10]+400*digital[11]+300*digital[12]+200*digital[13]+100*digital[14]+0*digital[15]);}
  suma=(digital[0]+digital[1]+digital[2]+digital[3]+digital[4]+digital[5]+digital[6]+digital[7]+digital[8]+digital[9]+digital[10]+digital[11]+digital[12]+digital[13]+digital[14]+digital[15]);
  pos=sumap/suma;
  if(poslast<=100 && pos==-1){
    pos=0;
```

```
}
if(poslast>=1400 && pos==-1){
  pos=1500;
}
poslast=pos;
return pos;
}

void PID(){
  proporcional=pos-setpoint;
  derivativo=proporcional-last_prop;
  integral=error1+error2+error3+error4+error5+error6;
  last_prop=proporcional;
  error6=error5;
  error5=error4;
  error4=error3;
  error3=error2;
  error2=error1;
```

```
  error1=proporcional;
  int diferencial=(proporcional*KP) + (derivativo*KD) + (integral*KI);
  if(diferencial > vel) diferencial=vel;
  else if(diferencial < -vel) diferencial=-vel;
  (diferencial < 0)?
  motores(vel, vel+diferencial):motores(vel-diferencial, vel);
}

void frenos(){
  if(pos<=250){
    motores(veladelante, -velatras);
  }
  if(pos>=1350){
    motores(-velatras, veladelante);
  }
}
void motores (int izq, int der){ //VELOCIDAD REGULADA DESDE 0 HASTA 255 o 0 hasta -255
```

```
////////////////MOTOR IZQUIERDO/////////////
if(izq>=0){
  digitalWrite(izq1, HIGH);
  digitalWrite(izq2, LOW);
}
else{
  digitalWrite (izq1, LOW);
  digitalWrite (izq2, HIGH);
  izq=izq*(-1);
}
analogWrite(pwmi,izq);

////////////////MOTOR DERECHO///////////////

 if(der>=0){
  digitalWrite(der1, HIGH);
  digitalWrite(der2, LOW);
}
```

```
  else{
    digitalWrite (der1, LOW);
    digitalWrite (der2, HIGH);
    der=der*(-1);
  }
  analogWrite(pwmd,der);
}
```

CAPITULO VI IMPLEMENTACION DEL SEGUIDOR DE LINEA

Con el seguidor ensamblado, el programa finalizado y cargado en el Arduino Nano solo resta explicar cómo activar el seguidor para su funcionamiento en pista. Para esto se debe colocar al seguidor encima de la pista a recorrer, se debe tener conectada la batería que suministrará energía y con esto se deben subir ambos Switch para que tanto la turbina como el Arduino Nano enciendan. Para comenzar con la calibración lo primero que se debe hacer es colocar al seguidor sobre el fondo completamente, principalmente por los sensores tomando que la programación debe estar adecuada al fondo y a la línea. Una vez el seguidor sobre el fondo se debe dar click a pulsador para que capture la información del fondo por medio de los sensores.

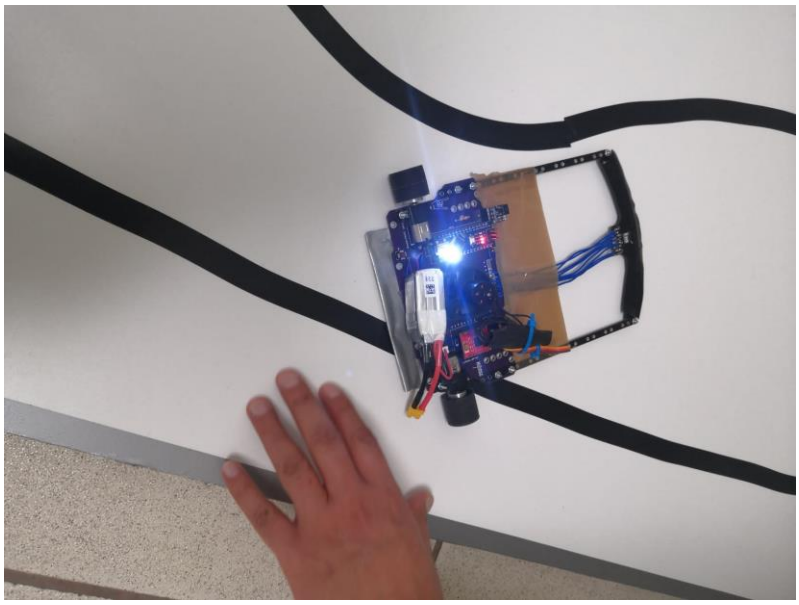


Imagen 257. Seguidor calibrando en fondo.

Después se debe realizar una segunda calibración sobre la línea de la pista, colocando toda la barra de sensores sobre la línea en medida de lo posible apoyándose principalmente de una curva para que todos los sensores capturen la línea.

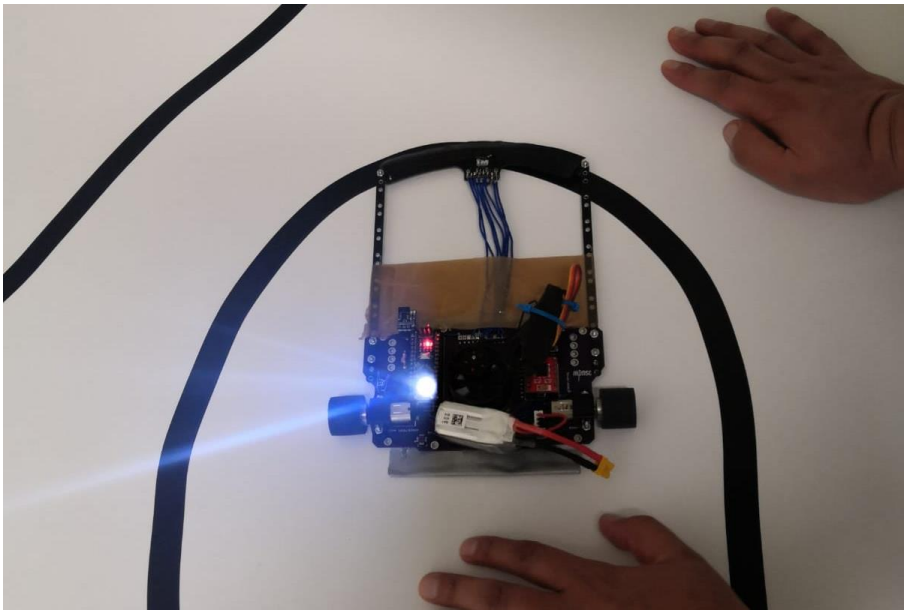


Imagen 258. Seguidor calibrando en linea.

Con la calibración de los sensores lo siguiente es presionar una vez más el pulsador para que la turbina encienda en su punto de espera, o sea no su velocidad máxima sino una velocidad menor para que el encendido no sea abrupto, después se debe presionar una última vez al botón pulsador para que la turbina en este momento alcance su máxima velocidad lista para hacer succión sobre la pista y ayudar al agarre del seguidor en su trayecto.

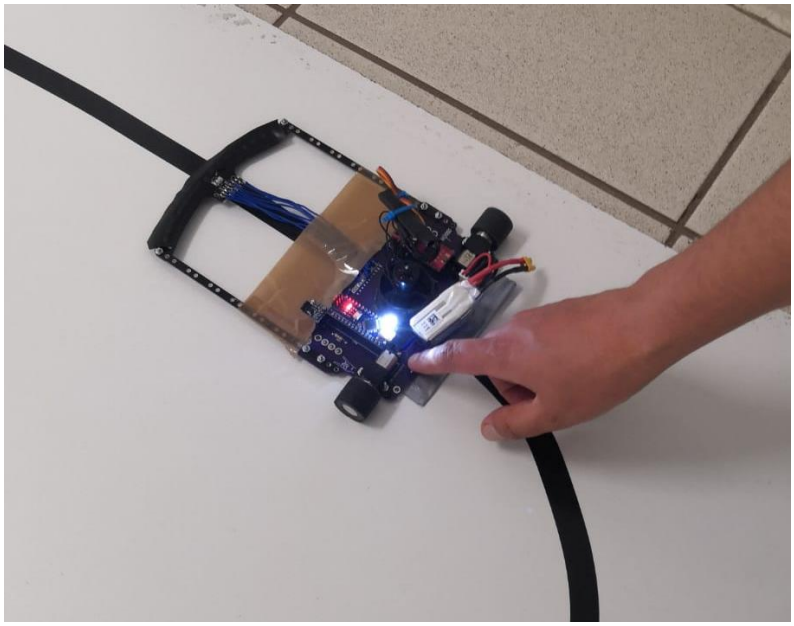


Imagen 259. Seguidor pulsando el botón para la turbina en reposo y para velocidad máxima.

Para que el seguidor inicie con su arranque por medio del control de juez primero se debe presionar el botón RESET, seguido del botón READY que encenderá un led rojo en el módulo arrancador dando paso a presionar por último el botón GO que hace correr al seguidor sobre la pista, muestra de ello es que en el módulo arrancador se enciende un led verde. Cuando se deba detener el seguidor únicamente se debe dar RESET en el control de juez.

Imagen 260. Control de Juez de Ingeniero Maker.

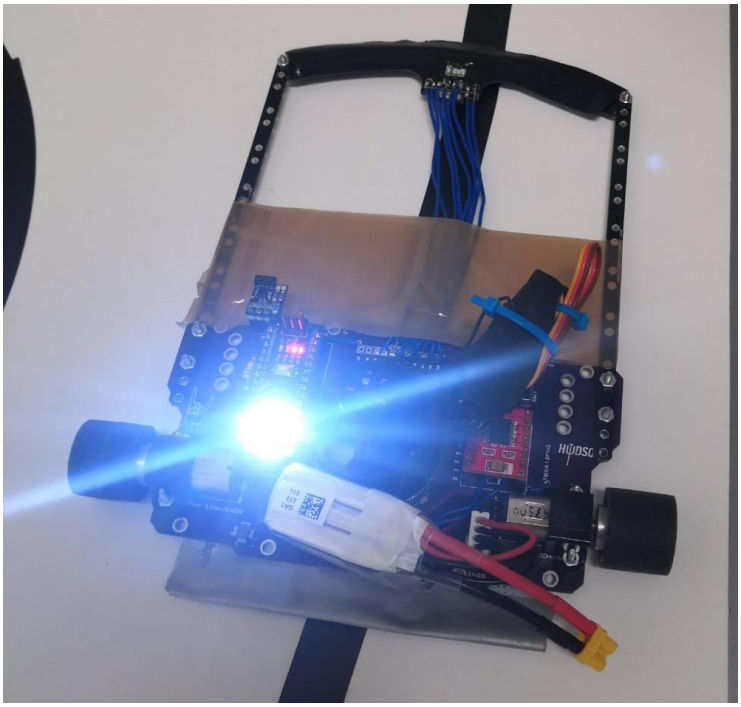


Imagen 261. Seguidor de línea sobre pista.

Conclusión

En resumen, el controlador PID de orden fraccionario representa una extensión poderosa y flexible del controlador PID convencional al permitir una respuesta más precisa y adaptable en sistemas con comportamientos no lineales y dinámicos complejos. Su capacidad para ajustar la constante de tiempo fraccional ofrece una mayor versatilidad para abordar fenómenos como el tiempo muerto y la inercia variable en sistemas de control. Aunque su implementación puede ser más desafiante debido a la naturaleza matemática y la selección adecuada de parámetros, el controlador PID de orden fraccionario demuestra su utilidad en aplicaciones donde la precisión y la adaptabilidad son esenciales para lograr un rendimiento óptimo del sistema.

Bibliografía

- Tepljakov, A., Baykant, B., Yeroglu, C., Gonzalez, E., Hossein Nia, S., Petlenkov; (2018), FOPID CONTROLLERS AND THEIR INDUSTRIAL APPLICATIONS: A SURVEY OF RECENT RESULTS. University of Technology, Estonia; Department of Computer Engineering, Malatya, Turkey.

- Naifar, O., Ben Makhlouf, A., (2021), FRACTIONAL ORDER SYSTEMS – CONTROL THEORY AND APPLICATIONS. Warsaw, Poland. Polish Academy of Sciences.

- Vázquez, U., (2022), ANÁLISIS DEL DESEMPEÑO DE UN CONTROL PID DE ORDEN FRACCIONAL EN UN ROBOT MÓVIL DIFERENCIAL. México. Revista Iberoamericana de Automotica e Infomormatica Industrial. Pag 74-83.

- Estévez Pérez, S., León Gil, J., Matesanz García, A., (2015), APLICACIÓN DE TÉCNICAS DE CONTROL FRACCIONARIO Y COMPARACIÓN DE RESULTADOS CON RESPECTO AL CONTROL CLÁSICO. México. Universidad de la Laguna.

- Reza Faieghi, M., Nemati, A., (2015), ON FRACTIONAL-ORDER PID DESIGN. Iran. Department of Electrical Engineering, Islamic Azad University.

- Viola Villamizar, J., (2012), CONTROL FRACCIONARIO APLICADO AL DISEÑO DE CONTROLADORES. Colombia. Escuela de Ingeniería, Universidad Pontificia Bolivariana.

- Tepljakov, A., (2011), FRACTIONAL – ORDER CALCULUS BASED IDENTIFICATION AND CONTROL OF LINEAR DYNAMIC SYSTEMS. Estonia. Tallinn University of Technology.

- Vinagre, M., Monje A., (2006), INTRODUCCIÓN AL CONTROL FRACCIONARIO. España. Escuela de Ingenierías Industriales, Universidad de Extremadura.

- Hernández Gaviño, R., (2010), INTRODUCCIÓN A LOS SISTEMAS DE CONTROL: CONCEPTOS, APLICACIONES Y SIMULACIÓN CON MATLAB. México. Primera Edición. Editorial Pearson Educación.

- Kacprzyk, J., (2013) INTELLIGENT FACTIONAL ORDER SYSTEMS AND CONTROL. AN INTRODUCTION. India. First Edition. Springer Editorial.