Universidad Autónoma Metropolitana

Casa abierta al tiempo Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería Electrónica

Modalidad: Proyecto Tecnológico.

“Diseño y construcción de un prototipo mecatrónico inteligente de iluminación en ambientes festivos”

Alumno: Michaca España Gadiel Moavi

Matricula: 205360839

Asesor: Jaimes Ponce Jorge Miguel

Co-asesor: Pérez Moreno Romy

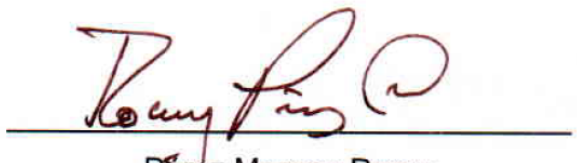
Trimestre 16-P

Fecha: 22 de septiembre de 2016

Yo, Jaimes Ponce Jorge Miguel, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Jaimes Ponce Jorge Miguel

Yo, Pérez Moreno Romy, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Pérez Moreno Romy

Yo, Gadiel Moavi Michaca España, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Gadiel Moavi Michaca España

Resumen

En este documento se describen los procedimientos realizados para el diseño y la construcción de un prototipo mecatrónico inteligente de iluminación, divididos en varios capítulos, empezando con la investigación de antecedentes específicos que fuesen de utilidad para definir las características generales de diseño, materiales a utilizar, características mecánicas, factores involucrados en el proceso de maquinado y ensamblado del sistema mecánico.

Se describen las consideraciones teóricas para llevar a cabo el diseño de los distintos componentes, que son de gran utilidad para definir sus características físicas y mecánicas.

En este documento también se describen los distintos bloques y etapas que conforman el sistema electrónico, como la etapa de comunicación serie-usb en la cual se contemplan las conexiones a realizar para adecuar las señales y las configuraciones para realizar un intercambio de información eficiente y sin perdidas de datos. De igual manera se detalla la teoría básica sobre diseño de filtros activos de segundo orden, para la obtención de ecuaciones que son de utilidad para el cálculo de los valores de componentes que los integran y que nos permiten obtener dicho comportamiento.

Se incluye información básica sobre el funcionamiento de los motores a pasos que son primordiales para el desarrollo de este proyecto al igual que diodos LED y los modulos de comunicación inalámbrica.

En el desarrollo del proyecto se incluye información sobre la manufactura de las piezas del prototipo, los materiales utilizados y el herramental utilizado, de la misma manera se encuentra información sobre las simulaciones de los circuitos electrónicos, graficas de respuesta en frecuencia de los filtros y la fabricación de las placas de circuito impreso PCB.

Índice de contenido

1. Introducción

El presente trabajo ejemplifica el trabajo y todos los factores que conlleva la realización de un sistema mecatrónico, como el desarrollo e implementación de nuevos sistemas que faciliten y mejoren la funcionalidad y/o movilidad de ciertas partes reformando y actualizando su diseño.

La inclusión de nuevas tecnologías electrónicas es otra parte importante que se considera al desarrollar un prototipo, ya que permite explorar las posibilidades y los alcances de aplicación en nuevos sistemas, mejorándolos y ampliando sus aplicaciones.

El desarrollo de este proyecto es con la finalidad de ofrecer opciones nuevas, prácticas y novedosas al mundo del entretenimiento e iluminación, brindando la opción de liberar tareas con la constate operación y monitoreo de dichos sistemas, a la vez que elimina el uso de cables para su control, que con el tiempo sufren un deterioro por su manipulación, transporte y desgaste de conectores lo que implica la aparición de fallas inesperadas, que si no se cuenta al momento con el reemplazo adecuado no podrá ser solucionada dicha situación.

2. Antecedentes

Para evolucionar hacia sistemas de precisión, inteligentes y autónomos, la tecnología debe ser integrada con la mecatrónica para mejorar el funcionamiento y crear nuevas funciones [1].

La revolución electrónica ocurrida en los años 60 con la integración del transistor con otros dispositivos semiconductores en circuitos monolíticos permitió que en 1971 fuera inventado el microprocesador. Esto permitió transformar señales analógicas en digitales, permitiendo realizar cálculos y tomar decisiones basadas en los resultados del cómputo así como tomar acciones acordes a esas conclusiones y acumular conocimiento, datos e información en las memorias de las máquinas. Esta nueva funcionalidad dotó a las máquinas y sistemas con características tales como flexibilidad y adaptabilidad, acelerando el desarrollo de aplicaciones en el sector industrial [1].

En los años 80 la tecnología de los semiconductores creó los sistemas Micro-electromecánicos (MEMS) lo que condujo a una miniaturización de dispositivos y a nuevas dimensiones en las máquinas y sistemas [1].

Otra revolución tecnológica, conocida como Opto-electrónica, iniciada con el invento del Laser en 1960, hizo posible la creación de nuevos métodos de fabricación tales como los depósitos químicos por vapor, la epitaxis de haces moleculares y el enfocamiento de haces de iones para el micro-maquinado. Estos métodos permitieron la integración de elementos ópticos y electrónicos en componentes simples y compactos [1].

Pero antes de la existencia de los “Robots” ya había algunos antecedentes o intentos primitivos. Sin duda la primera luminaria móvil para espectáculos era muy similar a los Seguidores o Follow Spot. Es decir, un artefacto de luz montado en un pivote o trípode especial que permitiera a un operador movimientos horizontales y verticales, para mover el haz de luz, generalmente con la intención de “seguir” los pasos de un bailarín o actor. Pero no fue esa la primera vez que el hombre controló el movimiento de la Luz. Mucho más antiguos son los dispositivos de baliza giratoria (como la de los grandes Faros de Playa). Si bien la intención era totalmente distinta, sirve como ejemplo histórico [2].

Las estéticas de los Mega Shows, del Rock, del Musical, etc, nacidas en los años 80 (que aun hoy son paradigma), plantearon el uso de grandes cantidades de luminarias, utilizadas al mismo tiempo, en cambios de efectos y escenas muy veloces y complejas. La tecnología fue acompañando este criterio, y generó luminarias y controladores apropiados para tal fin. Es allí donde nace la idea de lograr que el haz de luz pudiera presentar movimientos automatizados en vivo y a la vista del espectador [2].

En la actualidad los sistemas mecatrónicos han evolucionado rápidamente en conjunto con el avance tecnológico, al integrar dispositivos con mejores características, como viene siendo un mayor procesamiento de datos a velocidades más elevadas y de menor tamaño en el caso de los procesadores

3. Justificación

En la actualidad se busca que los distintos dispositivos de uso cotidiano y/o especializado tengan una mayor autonomía y sean de fácil manejo, lo cual permite que desempeñen su función sin la constante supervisión humana.

Con este prototipo se busca innovar con características nuevas que no se encuentran comercialmente y brindar una mayor autonomía en la iluminación y animación de eventos festivos, con la transformación directa de audio musical para generar movimientos y secuencias lumínicas más acorde a los diversos ritmos existentes, lo cual garantice que no se repetirán las mismas secuencias de efectos cada cierto periodo de tiempo o durante un número determinado de ciclos con la implementación de algoritmos de procesamiento de audio para la generación de patrones luminosos y motrices.
En cuanto a características se pretende darle un valor agregado con la implementación de dispositivos de comunicación inalámbrica para la interacción con el prototipo, en el caso más especifico a través de una computadora personal (de escritorio o portátil) o un celular smartphone con sus respectivas plataformas.

4. Objetivos

Objetivo general

Diseñar y construir un prototipo mecatrónico con cabeza de iluminación móvil inteligente, autonomía en funciones, técnicas de control y configuración a través de las plataformas *Windows* y *Android*.

Objetivos particulares

- Desarrollar controladores basados en algoritmos de control inteligente para procesamiento de audio.
- Diseñar e implementar sistemas electrónicos de procesamiento, control y comunicación.
- Diseñar e implementar el sistema mecánico del dispositivo.
- Desarrollar un software para la interacción con el dispositivo.

5. Marco teórico

El desarrollo de un prototipo mecatrónico de este tipo representa un gran reto, debido a que requiere de amplios conocimientos en las tres aéreas principales involucradas, como lo son la Electrónica, Mecánica y Computación.

5.1. Características generales del prototipo

Primeramente se definió que la estructura principal debía estar conformada por una base capaz de brindar soporte y de contener la circuitería principal, además de una cabeza móvil con movimientos en dos ejes de forma horizontal y vertical, que a su vez contendrá los dispositivos de iluminación conocidos como diodos *LED*.

Tomando en cuenta la movilidad, las comunicaciones inalámbricas y los inconvenientes que presentan los cables por el deterioro en su manipulación, se considera que el prototipo debe ser controlado como un dispositivo esclavo, sin conexiones alámbricas y por medio de un dispositivo maestro a distancia.

Otra característica es la de poder conectar el dispositivo maestro a una computadora para su control y configuración, además de recibir y procesar audio para ejecutar secuencias de movimiento e iluminación, acordes al ritmo de forma autónoma para no requerir la constante supervisión de un operador dedicado.

Con el análisis realizado se identificaron dos bloques principales, el primero conformado por el sistema mecánico cuya importancia es de tal relevancia que sin este no se podrían realizar los movimientos deseados.

El segundo bloque corresponde al sistema electrónico, encargado de recibir, procesar información y realizar todas las tareas programadas del prototipo.
A continuación se detallan estos bloques principales.

5.2. Sistema mecánico

La importancia de este sistema, como se menciono antes, se deriva de la capacidad que le brinda a los dispositivos de realizar movimientos básicos y complejos, que van desde un grado de libertad hasta los requeridos para realizar dichos movimientos.

Para la realización de este prototipo se planteó la implementación de dos ejes conocidos como *tilt* y *pan*, movimientos rotatorios sobre los ejes horizontal y vertical

respectivamente, utilizando motores a pasos, poleas y bandas dentadas, y sistemas que permitan una rotación libre y optima para evitar esfuerzos en los motores por fricción o sobrecarga, implementados con rodamientos radiales y axiales.

Este sistema también abarca el diseño estructural, que conformará el cuerpo del prototipo, donde estarán sujetas todas las partes que lo conforman, esta estructura se contempla que será capaz de soportar tanto el peso de los componentes, como las fuerzas derivadas por los movimientos generados en la puesta en marcha de los motores o al trasladar el prototipo a distintos lugares.

Una vez analizadas todas las posibles situaciones involucradas con el sistema para su óptimo desempeño a largo plazo, principalmente con la resistencia mecánica de los materiales, es posible determinar los parámetros idóneos para realizar el diseño de cada una de las partes del prototipo.

5.3. Sistema electrónico

En este sistema, se encuentran todos los componentes electrónicos encargados de realizar las distintas funciones que han sido planteadas para este proyecto, el cual se divide en dos sistemas, cada uno encargado de realizar tareas distintas, pero con la diferencia de que uno de ellos conocido como esclavo, depende de las instrucciones y comandos que el sistema maestro le indique.

5.3.1. Sistema maestro

A este sistema se le atribuyen las funciones principales de operación como se muestra en la imagen 5.1.

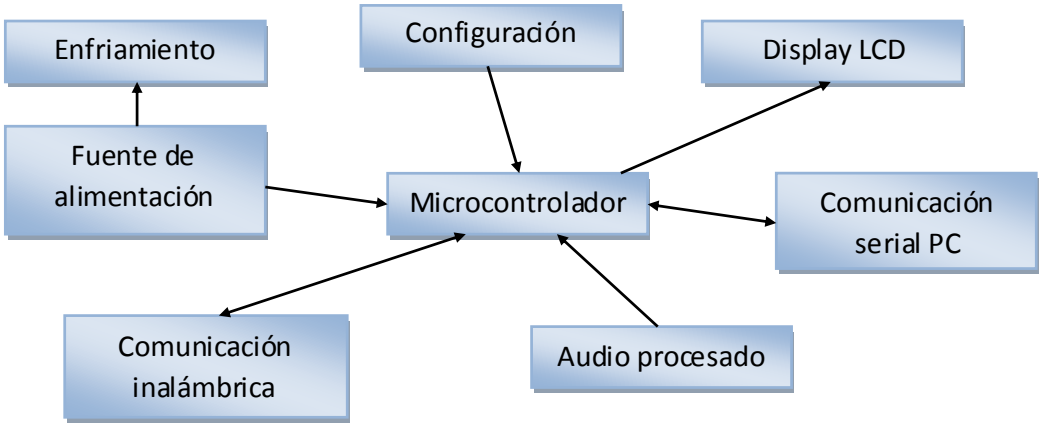


Figura 5.1. Diagrama a bloques del sistema maestro.

Con base en los análisis realizados se plantea realizar un diseño modular de las etapas principales del sistema para facilitar el reemplazo de las mismas en caso de presentarse fallas en los circuitos electrónicos.

- **Microcontrolador**

El microcontrolador es un chip de Silicio en cuyo interior posee la misma arquitectura de un computador personal, este necesita ser programado para realizar una simple función como el parpadeo de un *LED*, hasta un sofisticado sistema de control automatizado de una fábrica [3].

Es la parte principal de este sistema, encargado de realizar todas las operaciones y procesos referentes a la comunicación, tanto con la *PC* como con el microntrolador esclavo, visualización de los datos y el procesamiento de audio.

- **Fuente de alimentación y enfriamiento**

En este bloque se plantea el uso de una fuente de alimentación dual para el suministro de voltajes positivos y negativos primordiales en la implementación de Amplificadores Operacionales (*OPAMP*) [4], el Microcontrolador y otros dispositivos que operan con niveles de voltaje lógicos, sin olvidar el sistema de enfriamiento desarrollado con un ventilador para mantener los circuitos a temperaturas optimas por prolongados tiempos de operación.

- **Configuración y visualización**

Como se aprecia en el diagrama, también se plantea la implementación de un bloque de configuración con *DiP Switches*, el cual permite elegir de forma directa y manual el modo en el que operará el sistema maestro, en el caso de no disponer de una conexión Serial-USB con un equipo de cómputo. El modo de conexión o configuración se mostrará en un display LCD 16x2 donde el usuario podrá apreciar los estados de operación actual del prototipo.

- **Comunicación serial PC** [5]

Para realizar la comunicación con una computadora se utiliza el puerto *USART* del *AVR*, que viene de receptor transmisor síncrono asíncrono universal, es una forma de comunicación entre dispositivos que tengan esta capacidad, donde los datos pueden ser

enviados en grupos de 5, 6, 7, 8 o de 9 bits pero bit por bit, esto es en serie, por eso se dice que esta es una comunicación serial.

Para la comunicación serial asíncrona entre microcontroladores y para la comunicación entre el microcontrolador y el ordenador, se necesitan 2 hilos de conducción para la transmisión y recepción de datos, y un hilo de conducción para la conexión de los comunes o *GND* que tienen que ser los mismos, para la comunicación serial entre el microcontrolador y el ordenador se seguirá la norma *RS232*.

En la comunicación *USART AVR* asíncrona, uno de los hilos será para la transmisión de los datos de un dispositivo a otro y el otro hilo será para la recepción de datos entre un dispositivo a otro, la transmisión y la recepción pueden ocurrir en forma simultánea, lo que si se tiene que cumplir es que la frecuencia de trabajo de ambos dispositivos tiene que ser la misma, a esto se le conoce como los *baudios* que viene a ser la cantidad de bits por segundo que se transmitirán entre ambos dispositivos.

Los niveles de tensión con los que trabajan los pines del módulo *USART AVR* son de *0V* y *5V* un bajo será *0V* mientras que un alto será *5V*, por eso cuando la comunicación es entre microcontroladores la conexión entre pines se puede hacer directamente, pero cuando la comunicación es entre el microcontrolador y un ordenador la conexión entre pines tiene que hacerse a través de un conversor de nivel como el *MAX232*, ya que los niveles de tensión para la comunicación serial del ordenador son mayores que para el *AVR* típicamente entre *-12V* y *12V*, además de trabajar con lógica negativa, esto es para el ordenador un bajo será *12V* mientras un alto será *-12V*.

El socket donde se conectará el *AVR* con el ordenador para la comunicación serial es del tipo *DB9* y se conoce como puerto serie, pero resulta que este tipo de puerto ya no viene en los ordenadores portátiles que son los que hoy en día la mayoría utiliza, lo que traen ahora son puertos *USB*, por lo que para realizar la comunicación serial con el módulo *USART AVR* será necesario la utilización de un conversor *SERIE-USB* como el que se muestra en la siguiente imagen.

Imagen 5.2. Adaptador Serie-USB

El conector *DB9* es el que se utiliza para la comunicación serial con el módulo *USART AVR*, este conector consta de de 9 pines, los hay hembra y macho, de este conector solo se utilizarán 3 pines, uno para el pin *RX*, otro para el pin *TX* y el otro para el tierra, estos pines tiene una numeración que hay que respetar.

Imagen 5.3. Conectores DB9

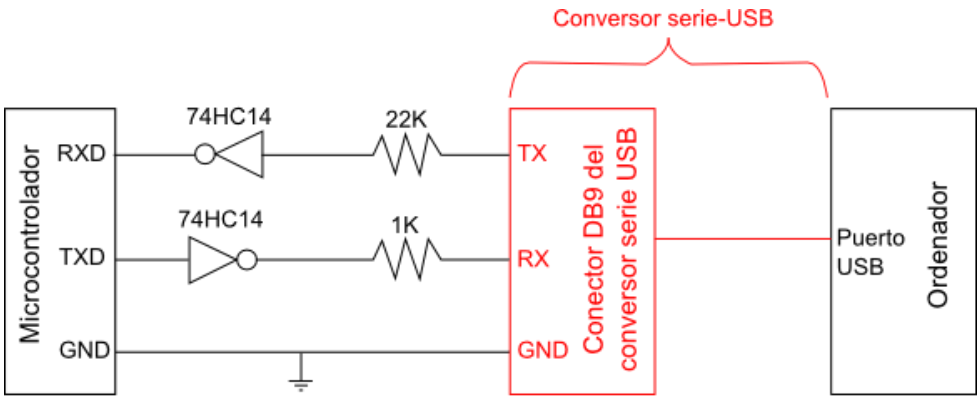
El conversor *Serie-USB* viene con el conector *DB9* macho, el cual se conecta al microcontrolador *AVR* ya sea a través del *MAX232* o con el uso de la compuerta inversora como se comento líneas arriba, por lo que los pines a conectar del *DB9* macho con el módulo *USART AVR* son en el siguiente orden:

- El pin 2 que es pin receptor *RX* del conector *DB9* macho que viene del ordenador, se conectará a través del *MAX232* o con el uso de la compuerta inversora y la resistencia de 1K al pin transmisor o pin *TXD* del microcontrolador *AVR*.
- El pin 3 que es pin transmisor *TX* del conector *DB9* macho que viene del ordenador, se conectará a través del *MAX232* o con el uso de la compuerta

inversora y la resistencia de 22K al pin receptor o pin *RXD* del microcontrolador *AVR.*

- El pin 5 que es pin *GND* o tierra o común del conector *DB9* macho que viene del ordenador, se conectará al pin *GND* del microcontrolador *AVR.*

Las conexiones del módulo *USART AVR* con el ordenador a través de la compuerta inversora y las resistencias, con el uso del conversor *Serie-USB* serán como se indica en la siguiente imagen.



Comunicación entre el microcontrolador AVR y el Ordenador con la compuerta inversora 74HC14 utilizando el conversor serie-USB, en la comunicación asíncrona del módulo USART AVR

Figura 5.4.

- **Procesamiento de audio**

En este bloque se integran todos los procesos de acoplamiento, filtrado y acondicionamiento de la señal de audio para poder ser procesada por el microcontrolador y así generar los comandos de movimientos y encendido de *LEDs* que serán enviadas al sistema esclavo.

En la figura 5.5 se muestra el diagrama del proceso planteado para realizar dicha tarea.

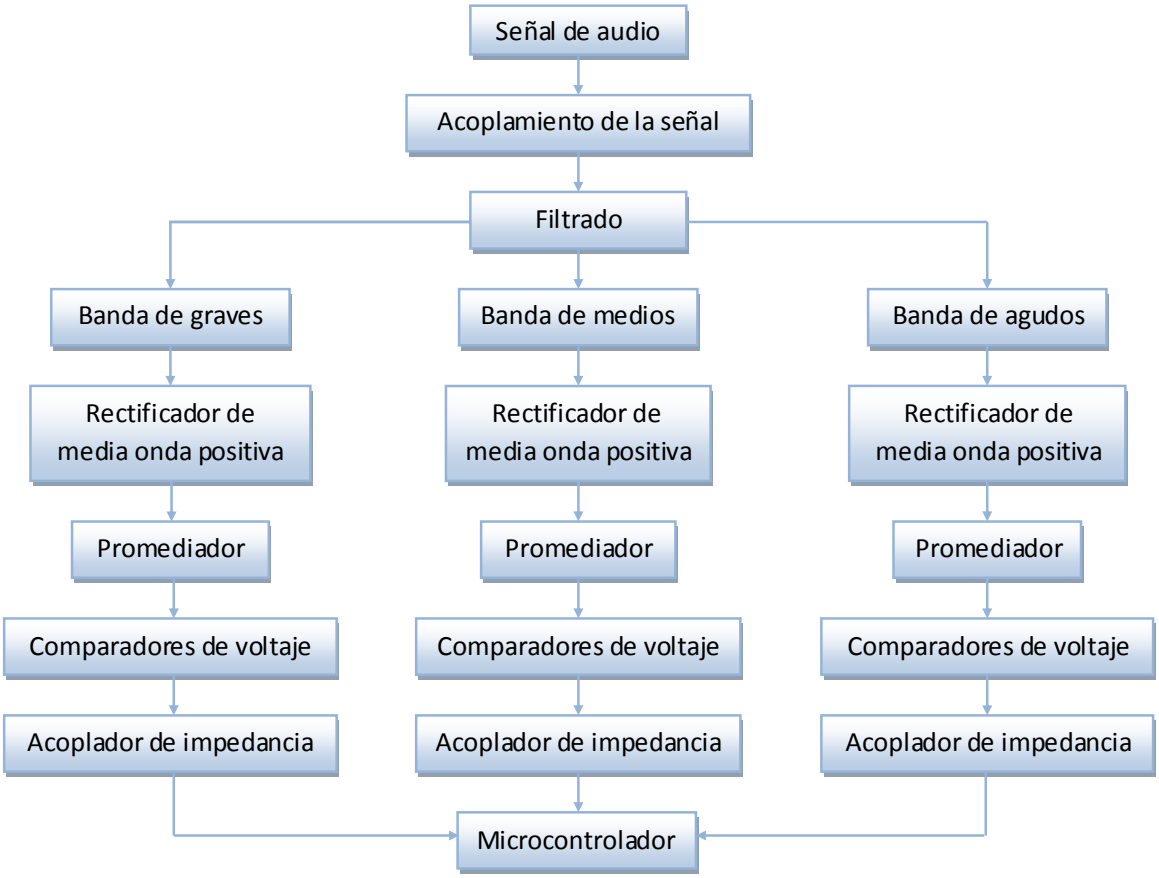


Figura 5.5. Diagrama de bloques del procesamiento de audio.

En ingeniería, se conocen como filtros a los sistemas selectivos de frecuencias que permiten el paso de una banda específica, mientras bloquean, o por lo menos atenúan, los componentes espectrales que se encuentran por fuera de este intervalo [6]. Los filtros pueden construirse a partir de la interconexión de resistencias, inductancias y condensadores, a los cuales se les conoce como pasivos; o a través de circuitos basados en amplificadores operacionales, en cuyo caso se denominan filtros activos.

Una de las principales ventajas que ofrecen los filtros activos consiste en que son ideales para operar a bajas frecuencias ya que no involucran el uso de los incómodos inductores, además de aprovechar el predecible comportamiento que poseen los amplificadores operacionales en las regiones inferiores del espectro. Esto hace que este tipo de sistemas sean sumamente versátiles a la hora de procesar señales cuyo ancho de banda se sitúa dentro de la audio frecuencia, región en la cual los filtros pasivos presentan algunas dificultades, pues los inductores requeridos suelen ser de alta capacidad y físicamente

voluminosos, además de demandar corrientes relativamente altas a las fuentes de tensión que producen la señal a filtrar [7].

Filtro pasa bajos activo de segundo orden

Del análisis de filtros pasivos se define la función de transferencia de un selector de bajas frecuencias de segundo orden mediante la expresión:

$$G(s) = \frac{\omega_c^2}{s^2 + \frac{\omega_c}{Q}s + \omega_c^2} \quad (1)$$

Donde _c representa la frecuencia de corte del filtro y Q el factor de calidad del mismo [8]. Considérese ahora el circuito mostrado en la figura 1, el cual se conoce como red de Sallen-Key, en el que Y1 a Y5 representan admitancias que pueden proceder de resistencias o condensadores [9].

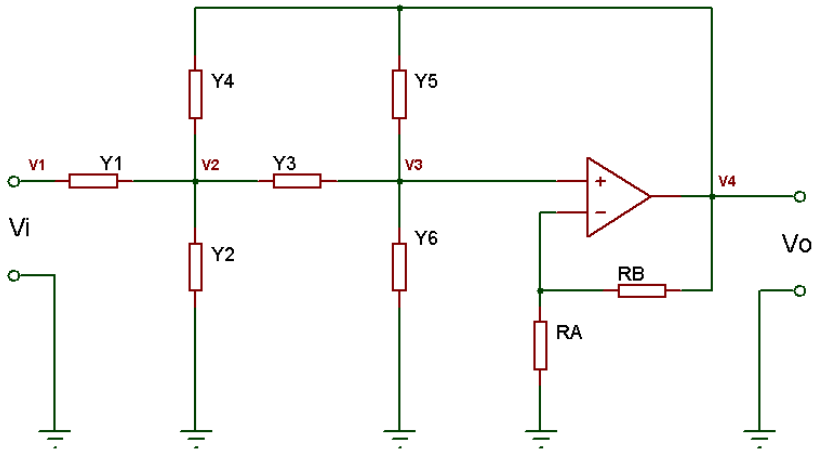


Figura 5.6. Red Sallen-Key.

En el circuito anterior se identifica una subred conformada por un amplificador operacional en configuración no inversora, la cual, para simplificar los cálculos, puede expresarse en términos de su ganancia, denominada μ, de tal forma que:

$$v_0 = \left(1 + \frac{R_B}{R_A}\right)v_3 \quad (2) \qquad v_4 = \left(1 + \frac{R_B}{R_A}\right)v_3 \quad (3) \qquad v_4 = \mu v_3 \quad (4)$$

Si la expresión de ganancia de tensión correspondiente al circuito de la figura 1 es comparable a la ecuación (1), es posible afirmar entonces que el circuito en cuestión se comporta como un filtro activo pasa bajos de segundo orden. Para la obtención de dicha función de transferencia es viable aplicar el modelo de descripción matricial de redes [10], con lo cual se obtiene la siguiente expresión:

$$\begin{bmatrix} i_i \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} Y_1 & -Y_1 & 0 & 0 \\ -Y_1 & Y_1+Y_2+Y_3+Y_4 & -Y_3 & -Y_4 \\ 0 & -Y_3 & Y_3+Y_5+Y_6 & -Y_5 \\ 0 & -Y_4 & -Y_5 & Y_4+Y_5 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \quad (5)$$

Introduciendo la ecuación de restricción (4) en la matriz anterior, se obtiene:

$$\begin{bmatrix} i_i \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} Y_1 & -Y_1 & 0 & 0 \\ -Y_1 & \alpha & -Y_3 & -Y_4 \\ 0 & -Y_3 & \beta & -Y_5 \\ 0 & -Y_4 & -Y_5 & \gamma \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \mu v_3 \end{bmatrix} \quad (6)$$

Donde:

$$\alpha = Y_1 + Y_2 + Y_3 + Y_4 \quad (7)$$

$$\beta = Y_3 + Y_5 + Y_6 \quad (8)$$

$$\gamma = Y_4 + Y_5 \quad (9)$$

A partir de la matriz (6) se obtienen las ecuaciones:

$$i_i = Y_1 v_1 - Y_1 v_2 \quad (10)$$

$$0 = -Y_1 v_1 + \alpha v_2 - Y_3 v_3 - Y_4 \mu v_3 \quad (11)$$

$$0 = -Y_3 v_2 + \beta v_3 - Y_5 \mu v_3 \quad (12)$$

$$0 = -Y_4 v_2 - Y_5 v_3 + \gamma \mu v_3 \quad (13)$$

Las expresiones anteriores conforman un sistema lineal de 4 ecuaciones y 3 incógnitas, por lo cual solo se toman las 3 primeras. De esta forma la matriz queda (6) se transforma en:

$$\begin{bmatrix} i_i \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} Y_1 & -Y_1 & 0 \\ -Y_1 & \alpha & [-(Y]_3 + \mu Y_4) \\ 0 & -Y_3 & \beta - \mu Y_5 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad (14)$$

Aplicando los conceptos de descripción matricial de redes, la ganancia de tensión de toda la red está definida como:

$$A_V = \frac{v_0}{v_i} = \frac{v_4}{v_1} = \frac{\mu v_3}{v_1} = \mu \frac{Y_3^1}{Y_1^1} * \frac{(-1^{3+1})}{(-1^{1+1})} \quad (15)$$

Donde los términos Y_3^1 y Y_1^1 representan los determinantes de las matrices que resultan al eliminar fila 1 columna 3 y fila 1 columna 1 de la expresión (14), respectivamente. Según lo anterior:

$$Y_3^1 = \Delta \begin{vmatrix} -Y_1 & \alpha \\ 0 & -Y_2 \end{vmatrix} = Y_1 Y_2 \quad (16)$$

Y

$$Y_1^1 = \Delta \begin{vmatrix} \alpha & -(Y_3 + \mu Y_4) \\ -Y_3 & \beta - \mu Y_3 \end{vmatrix} = \alpha\beta - \mu Y_5 \alpha - Y_3^2 - \mu Y_3 Y_4 \quad (17)$$

Reemplazando (16) y (17) en (15):

$$A_V = \frac{\mu Y_1 Y_2}{\alpha\beta - \mu Y_5 \alpha - Y_3^2 - \mu Y_3 Y_4} \quad (18)$$

Involucrando las expresiones (7), (8) y (9) en (18), se llega finalmente a la siguiente ganancia de tensión para la red Sallen-Key de la figura 1:

$$A_v = \frac{\mu Y_1 Y_2}{(Y_1 + Y_2 + Y_4)(Y_3 + Y_5 + Y_6) + Y_3(Y_5 + Y_6) - \mu[Y_2 Y_4 + Y_5(Y_1 + Y_2 + Y_3 + Y_4)]} \quad (19)$$

Como se dijo anteriormente, la expresión anterior debe compararse con la ecuación (1), de tal forma que la red pueda considerarse como un filtro pasa bajos activo de segundo orden. Para lograr esta operación es posible afirmar que:

$$Y_1 = \frac{1}{R_1}; \quad Y_2 = \frac{1}{\infty}; \quad Y_3 = \frac{1}{R_2}; \quad Y_4 = C_4 S; \quad Y_5 = \frac{1}{\infty}; \quad Y_6 = C_6 S$$

Con lo cual la ganancia de tensión toma la forma:

$$A_V = \frac{\frac{\mu}{R_1 R_3}}{\left(\frac{1}{R_1} + C_4 S\right)\left(\frac{1}{R_3} + C_6 S\right) + \frac{1}{R_3}(C_6 S) - \mu\left(\frac{C_4 S}{R_3}\right)}$$

$$A_V = \mu \frac{\frac{1}{R_1 R_3 C_4 C_6}}{S^2 + \left(\frac{1}{R_1 C_4} + \frac{1}{R_3 C_6} + \frac{1}{R_3 C_4} - \frac{\mu}{R_3 C_6}\right) S + \frac{1}{R_1 R_3 C_4 C_6}} \quad (20)$$

Al comparar (20) con (1), se concluye que:

$$\omega_C^2 = \frac{1}{R_1 R_3 C_4 C_6} \quad y \quad \frac{\omega_C}{Q} = \left(\frac{1}{R_1 C_4} + \frac{1}{R_3 C_6} + \frac{1}{R_3 C_4} - \frac{\mu}{R_3 C_6}\right)$$

$$\text{Si } R_1 = R_3 = R \text{ y } C_4 = C_6 = C :$$

$$\omega_C = \frac{1}{RC} \quad (21) \qquad \mu = 3 - \frac{1}{Q} \quad (22)$$

En resumen, para diseñar un filtro activo pasa bajos de segundo orden a partir de una red *Sallen-Key*, se elige un valor de capacitancia cualquiera y mediante la ecuación (21) se determina el valor de *R* que permite ajustar el valor de ω_C al deseado. De igual forma, se halla la ganancia de la red de amplificación no inversora μ, con el fin de ajustar el factor de calidad *Q*, el cual generalmente tiene un valor de *0,707*. Como el término μ multiplica la función de transferencia, la ganancia en la banda pasante del filtro será mayor que cero, por lo cual es necesario adicionar una red que permita normalizar esta amplificación que en algunos casos puede ser indeseada.

Filtro pasa altos activo de segundo orden

La función de transferencia de un sistema de segundo orden que permite el paso de componentes espectrales superiores a la frecuencia de corte ω_C, se puede escribir como:

$$G(s) = \frac{S^2}{s^2 + \frac{\omega_c}{Q} s + \omega_c^2} \quad (22)$$

Si se desea convertir la red Sallen-Key presentada en la figura 1 en un filtro pasa altos, es necesario comparar su función de transferencia (ver ecuación 18) con la expresión (22). Para esto es posible realizar las siguientes consideraciones:

$$Y_1 = C_1 S, \quad Y_3 = C_3 S, \quad Y_4 = \frac{1}{R_4}, \quad Y_2 = \frac{1}{\infty}, \quad Y_5 = \frac{1}{\infty}, \quad Y_6 = \frac{1}{R_6}$$

Con lo cual (18) se convierte en:

$$A_V = \frac{\mu C_1 C_3 S^2}{\left(C_1 S + \frac{1}{R_4}\right)\left(C_3 S6 + \frac{1}{R_6}\right) + \frac{C_3 S}{R_6} - \frac{\mu C_3 S}{R_4}} \quad (23)$$

$$A_V = \frac{\mu S^2}{S^2 + \left(\frac{1}{C_3 R_6} + \frac{1}{C_1 R_4} + \frac{1}{C_1 R_6} - \frac{\mu}{C_1 R_4}\right) S + \frac{1}{R_4 R_6 C_1 C_3}} \quad (24)$$

De forma análoga al filtro pasa bajos, si $R_6 = R_4 = R$ y $C_1 = C_3 = C$, se concluye que:

$$A_V = \mu \frac{S^2}{S^2 + \frac{3 - \mu}{RC} S + \frac{1}{RC}} \quad (25)$$

Al comparar (25) con (22) se obtiene:

$$\omega_C = \frac{1}{RC} \quad (26) \qquad \mu = 3 - \frac{1}{Q} \quad (27)$$

Así pues, para diseñar un filtro activo pasa altos de segundo orden basado en redes Sallen-Key, basta con intercambiar las impedancias utilizadas en el pasa bajos, es decir, sustituir las resistencias por capacitores y viceversa.

DISEÑO DEL FILTRO PASA BANDA

Para obtener un sistema que permita el paso de los componentes espectrales pertenecientes al intervalo $\omega_L - \omega_H$, con una atenuación inferior a *3dB*, es posible conectar

un filtro pasa bajos en cascada con un pasa altos, con frecuencias de corte configuradas en ω_H y ω_L respectivamente.

Para la implementación de la etapa del rectificador de media onda positiva y del Promediador, se consideró la función de un detector de pico que es la capturar el valor pico de la entrada y producir $Vo = Vi_{(pico)}$. Para alcanzar esta meta, se hace que *Vo* siga a Vi hasta llegar al valor pico. Entonces, este valor se retiene hasta que se presente un pico nuevo y más grande, en cuyo caso el circuito actualizará *Vo* al nuevo valor pico. En la figura 5.7a se muestra un ejemplo de formas de onda de entrada y de salida. Los detectores de pico tienen su aplicación en la instrumentación de pruebas y mediciones [4].

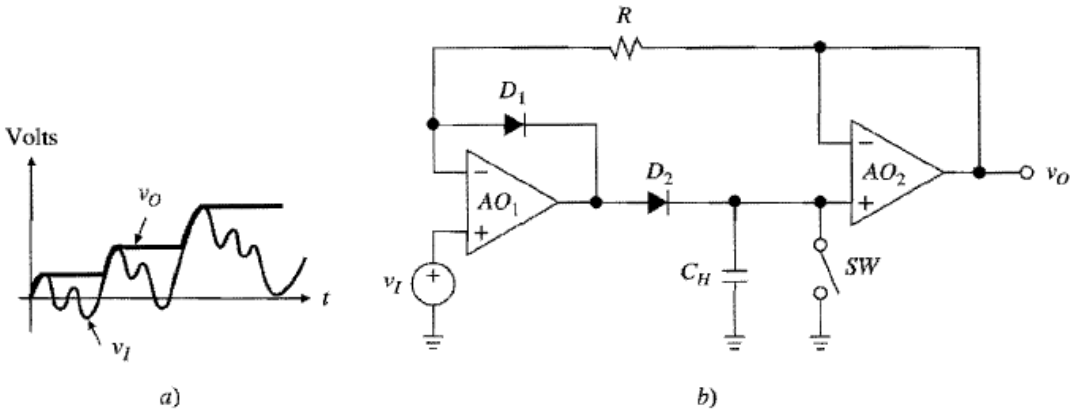


Figura 5.7. Formas de onda del detector de pico y diagrama del circuito.

Pero como se aprecia en la imagen, aun no se obtiene el comportamiento deseado, por lo tanto se agrega una resistencia en paralelo con el capacitor C_H, sustituyendo el switch SW, logrando de esta manera el comportamiento deseado que facilitará la obtención de las señales lógicas adecuadas para el acoplamiento con el microcontrolador, con la implementación de comparadores de voltaje referenciados.

Por último, antes de de que las señales lleguen al microcontrolador se implementa la etapa de acoplamiento de impedancia para asegurar el valor lógico “0” y evitar daños a los componentes involucrados.

- **Comunicación inalámbrica** [11]

El bloque de comunicación inalámbrica es implementado con el uso de un módulo transceiver que puede comunicarse con otro y ambos pueden enviar y recibir información,

al contrario de los módulos más económicos unidireccionales, estos módulos pueden enviar y recibir información al mismo tiempo.

El circuito integrado nRF2401+ está preparado para gestionar comunicación inalámbrica entre uno o varios módulos del mismo tipo, cuenta con buffers de entrada y salida, un modo de corrección de errores por auto retransmisión y varias características especiales. Se pueden comunicar varios módulos a la vez y hacer redes entre ellos.

La comunicación es por SPI lo que permite alcanzar velocidades de transmisión muy altas.

Para poder comunicar dos módulos entre si debe coincidir la dirección de 16bits además del canal de transmisión que se puede seleccionar entre 125 opciones, esto permite que la comunicación sea muy segura, además de que se puede seleccionar una dirección para transmitir y otra para recibir datos, lo que también permite crear varios tipos de redes en el mismo entorno sin perdidas de datos.

Este modulo es utilizado con el objetivo de operar y controlar las funciones del sistema esclavo a distancia, evitando inconvenientes comunes presentados por mal uso o constante manipulación del cableado en el caso del Protocolo DMX512 [12].

5.3.2. Sistema esclavo

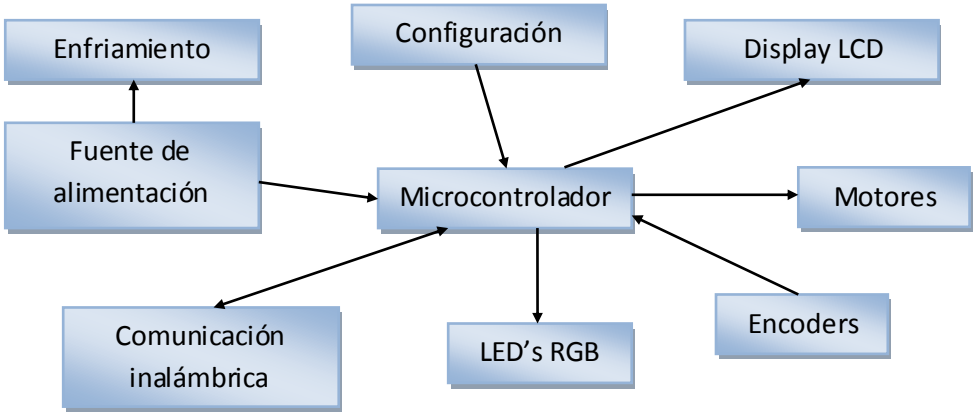


Figura 5.8. Diagrama del sistema esclavo.

- **Fuente de alimentación y enfriamiento**

Al igual que en el sistema maestro, en el esclavo tiene la función de proporcionar la energía necesaria para el funcionamiento de los componentes electrónicos, teniendo la diferencia de tener mayor potencia de operación por la integración de los motores a pasos y los diodos LED de potencia, siendo por estos últimos que se vuelve indispensable la implementación del sistema de enfriamiento para mantenerlos dentro del rango de temperatura de operación que va desde -35 a 60 °C o de lo contrario recibirán daños permanentes.

- **Configuración y visualización**

Este bloque presenta el mismo objetivo que su similar en el sistema maestro, que es el de configurar el dispositivo en modo dependiente o independiente de las instrucciones y comandos que se recibirán de forma inalámbrica, mostrando en el display dicha configuración.

- **Motores y encoders** [13][14]

Un motor paso a paso, como todo motor, es en esencia un conversor electromecánico, que transforma energía eléctrica en mecánica. Mientras que un motor convencional gira libremente al aplicarle una tensión, el motor paso a paso gira un determinado ángulo de forma incremental (transforma impulsos eléctricos en movimientos de giro controlados), lo que le permite realizar desplazamientos angulares fijos muy precisos (pueden variar desde 1,80º hasta unos 90º).

Los motores, tanto de corriente continua como de corriente alterna, son muy efectivos en muchas labores cotidianas desde la tracción de grandes trenes hasta el funcionamiento de lavarropas. Pero debido a problemas tales como la, inercia mecánica o su dificultad para controlar su velocidad, se desarrollaron otro tipo de motores cuya característica principal es la precisión de giro.

Existen algunos métodos para el control de este tipo de motores, según las secuencias de encendido de bobinas, en este caso se utiliza el método de medio paso, una combinación de los métodos de paso simple y paso doble, donde podemos hacer moverse al motor en pasos más pequeños y precisos, de esta forma obtenemos tenemos el doble de pasos de movimiento para el recorrido total de 360º del motor.

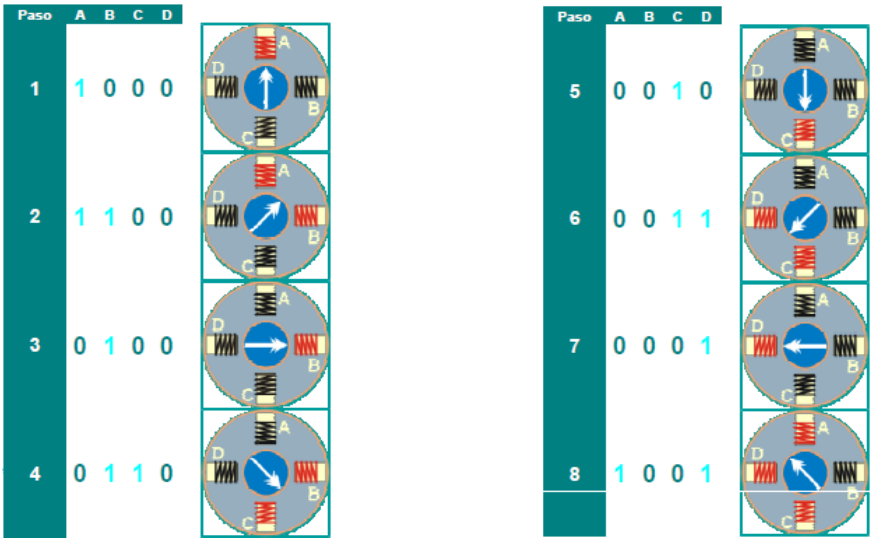


Figura 5.9. Movimientos por pasos de un motor “medio paso”

Los Encoders son sensores que generan señales digitales en respuesta al movimiento. Están disponibles en dos tipos, uno que responde a la rotación, y el otro al movimiento lineal.

Cuando son usados en conjunto con dispositivos mecánicos tales como engranes, ruedas de medición o flechas de motores, estos pueden ser utilizados para medir movimientos lineales, velocidad y posición.

Los encoders ópticos utilizan un disco de vidrio con un patrón de líneas depositadas en él, un disco metálico o plástico con ranuras (en un encoder rotatorio), o una tira de vidrio o metal (en un encoder lineal). La luz de un LED brilla a través del disco o tira sobre uno o más fotodetectores, que produce el suministrador del encoder. Un encoder de incremento tiene una o más de estas pistas, mientras que un encoder absoluto tiene varias pistas como bits de salida.

Figura 5.10. Tipos de encoders ópticos.

Estos dispositivos en conjunto son indispensables para realizar los movimientos del prototipo de forma adecuada y controlada, principalmente cuando las partes móviles están a punto de llegar al final de carrera o de giro.

- **Iluminación con diodos LED**

Los diodos emisores de luz (LEDs) son elementos de estado sólido (semiconductores) que emiten energía luminosa al aplicar energía eléctrica directamente, los cuales, dependiendo de la aplicación pueden ser de baja o alta potencia.

Los LEDs de alta potencia son diseños más completos que incluyen diversas alternativas ópticas de control del flujo luminoso y se fabrican en potencias mayores a 1 W; este tipo de LEDs se utilizan principalmente en aplicaciones arquitectónicas de iluminación, en exteriores e iluminación para calles, permitiendo tener más posibilidades de diseño y efectos de color [15].

Características:

-Vida promedio de 50,000 hrs.
-Excelente Flujo Luminoso
-Alta Eficiencia
-Control preciso y direccional del flujo luminoso
-Mínimas emisiones de radicaciones infrarrojas y ultravioletas

- **Comunicación inalámbrica**

Este bloque tiene la función de comunicarse con el sistema maestro, implementado con el mismo modulo módulo transceiver pero con la diferencia de que solo será capaz de recibir información, con el único propósito de realizar todas las tareas que se le indiquen.

6. Desarrollo del proyecto

6.1. Diseño mecánico y manufactura

Una vez definidas las características del sistema mecánico, se procedió a realizar los diseños de las distintas partes que lo integran, primeramente mediante bosquejos a mano con medidas aproximadas, tomando en cuenta dispositivos a utilizar que ya existen en el mercado y que sería más costoso e impráctico manufacturarlos, basándose en los datos y características contenidas en las hojas técnicas proporcionada por los distribuidores.

Al ser considerados y analizados todos los requisitos de diseño, se llevó a cabo el modelado del prototipo y sus distintos componentes a través de la plataforma Autodesk Inventor, considerando medidas reales y exactas para su futura manufacturación.

Los planos obtenidos del modelado de las piezas, con sus respectivas características y medidas se encuentran en el “Anexo A” al final de este documento.

Ya que fueron aprobados los diseños por los asesores, se procedió la adquisición del material, superando problemas de adquisición por inexistencia, materiales descontinuados, ajuste y rediseño de piezas para posteriormente solicitar la asesoría de los técnicos y el préstamo de equipo en los talleres de mecánica de la universidad para manufacturar las piezas.

Imagen 6.1. Material de aluminio disipador de calor

Imagen 6.2. Tubo de aluminio 1”

Imagen 6.3. Solera 4.8x19 mm

Para tal efecto se utilizó la siguiente maquinaria; Torno, Fresadora, Cizalla, Taladro de banco y manual, Cierra-cinta, y herramental como el Calibrador Vernier, Segueta, Maneral y Machuelos, lima metálica, entre otros, que fueron indispensables para el maquinado de las piezas, que en su mayoría se realizaron con el material de la imagen 6.1 y que se describen a continuación.

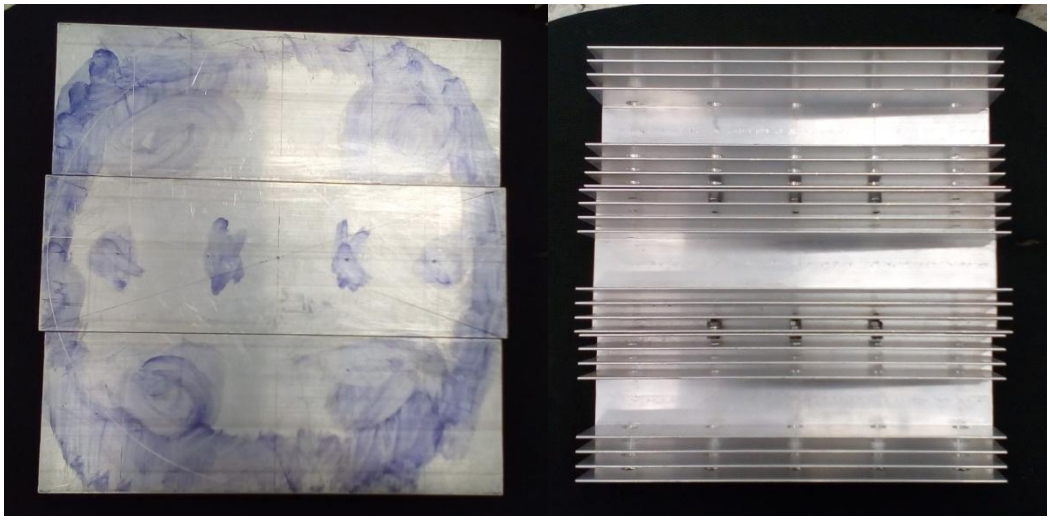


Imagen 6.3. Unión de tres barras disipadoras

Como se aprecia en la imagen 6.3, se unieron tres barras recortadas del disipador de calor para formar un cuadrado y de ahí poder cortar un disco de 234mm de diámetro,

donde serán colocados los LEDs y así mantener una temperatura adecuada de estos últimos.

Para realizar este corte se solicitó el apoyo de la área de Materializacion 3D de CyAD, donde se utilizó un Router CNC apoyados del siguiente bosquejo de la imagen 6.4.

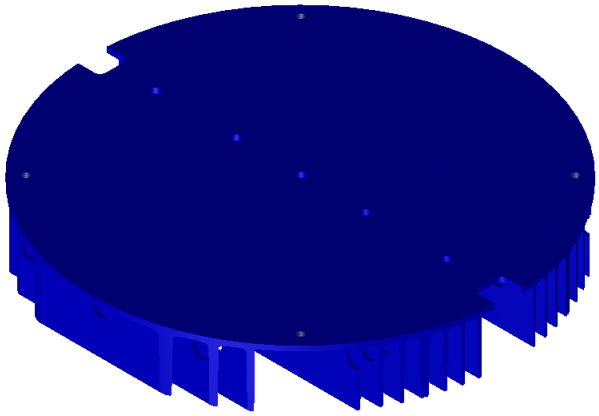


Imagen 6.4. Bosquejo del disco disipador de calor.

Utilizando el mismo material del disipador de calor, se manufacturó la base de la estructura de la cabeza móvil del prototipo, utilizando diversas herramientas, entre ellas la fresadora para desbaste de aletas y el torno para el barrenado central donde será acoplado el eje que brindará la capacidad de realizar los movimientos horizontales, el maquinado realizado se aprecia a continuación.

Imagen 6.5. Base de soporte de la cabeza móvil montada en el torno.

De igual manera los soportes laterales que contienen los componentes que forman el eje para los movimientos verticales, se maquinaron de la barra disipadora de calor con distintas herramientas y maquinas para lograr el resultado apreciando en las siguientes imágenes.

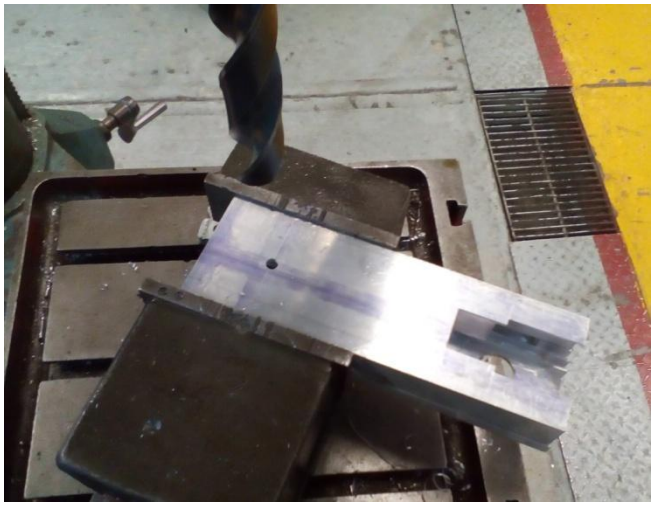


Imagen 6.6.Soporte antes de ser barrenado

Imagen 6.7 Barrenando el soporte

Imagen 6.8. Soporte después del barrenado

Imagen 6.9. Machuelado

En la imagen 6.9 se aprecia el machuelado de algunos barrenos para el atornillado y sujeción de los rodamientos que forman parte fundamental en los movimientos del eje vertical.

Para la implementación del eje vertical, como ya se mencionó antes, se utilizan rodamientos radiales con dimensiones de 1x2x0.5 in, sujetados por cajas de *Nylamid* maquinadas en el torno como se aprecia a continuación.

Imagen 6.10. Rodamientos acoplados en cajas de *Nylamid*.

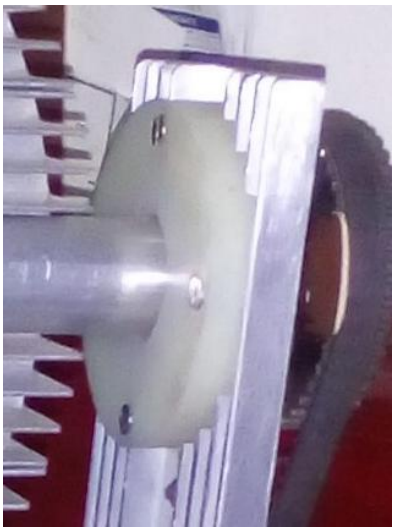


Imagen 6.11. Implementación del eje vertical.

Para la implementación de eje horizontal se diseño y maquinó una pieza capaz de fijarse a la base de la estructura principal del prototipo y que pudiera alojar los demás elementos que componen dicho eje, como son una polea dentada, dos rodamientos axiales con dimensiones de 1.575x2.77x0.748 in, y cuerda en los extremos para asegurar todas las partes con 4 tuercas en forma de sándwich.

Imagen 6.12. Algunas partes del eje horizontal.

Imagen 6.13. Partes del rodamiento axial.

Imagen 6.14. Parte principal del eje horizontal.

Para sujetar y darle firmeza a los soportes laterales de la cabeza móvil, y el disco disipador de calor, se maquinaron dos tipos sujetadores, unos con forma de “C” para sujetar el disco disipador de calor con el tubo que conforma el eje vertical y por último dos sujetadores largos para mantener la posición en escuadra de los soportes laterales con el de la base.

Imagen 6.15. Sujetadores

Para trasmitir el movimiento de los motores a cada uno de los ejes, se adquirieron bandas y poleas dentadas, estas últimas de distintos tamaños y por pares, una polea chica de 12 dientes que va colocada directamente en el motor y una de mayor dimensión que posee 42 dientes que va directamente sobre el eje, obteniendo una relación de giro de 3.5:1.

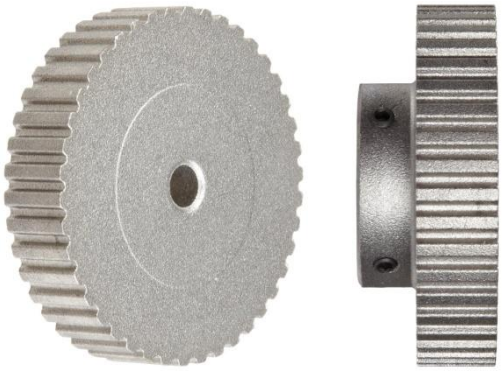


Imagen 6.16. Polea de hierro colado de 42 dientes “42XL037”

Imagen 6.17. Polea de hierro colado de 22 dientes “12XL037”

A la polea de la imagen 6.16 se tuvo que acondicionar acorde a las necesidades de los ejes ampliando el agujero central y realizando desbastes por la parte plana para reducir el peso de las mismas, a excepción de las poleas pequeñas (imagen 6.17) que se ajustaron a las características de los motores desde un principio.

A continuación se muestra el trabajo realizado en las poleas.

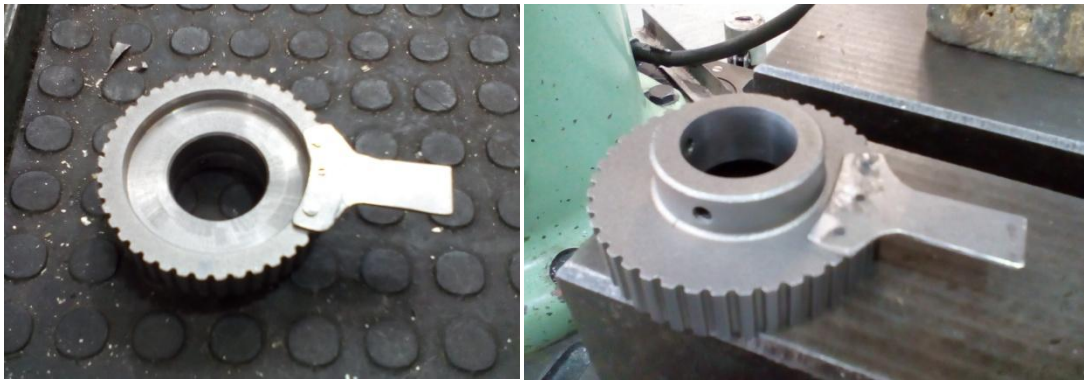


Imagen 6.18. Poleas acondicionadas.

Por último, se detalla una de las partes importantes, encargada de mantener acomodados los LEDs con sus respectivos lentes, en constante contacto con el disco disipador de calor.

Consta de una placa de acrílico de 5mm de grosor en forma circular, con un diámetro de 234mm y una segunda placa del mismo diámetro pero de 2mm de grosor, estos discos a su vez tiene 36 agujeros distribuidos geométricamente, con la diferencia que el diámetro de los agujeros de una placa difieren de la otra, pero con la cualidad de coincidir sus puntos centrales con su semejante a la hora de ser empalmadas, formando una pequeña caja circular para alojar el lente de los LEDs.

Esta pieza fue elaborada en la cortadora láser con el apoyo del área de Materialización 3D de CyAD.

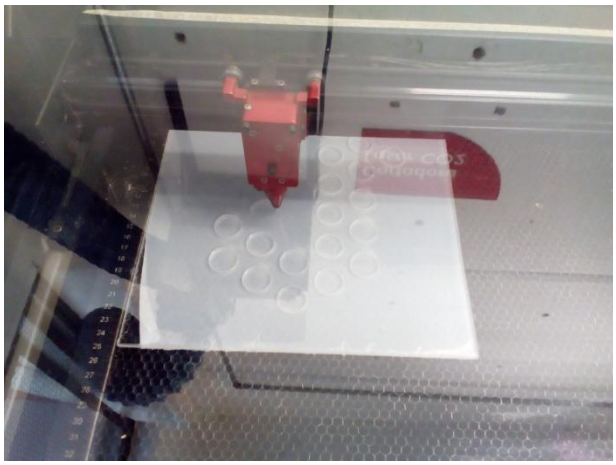


Imagen 6.19. Proceso de cortado con Láser de CO2

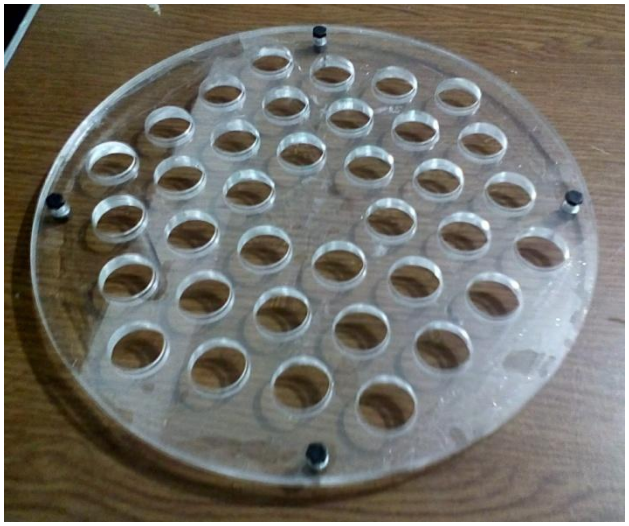


Imagen 6.20. Corte de acrílico terminado.

Imagen 6.20. Uso de la tapa de acrílico.

6.2. Diseño electrónico e implementación

En este capítulo se retomará la parte teórica del diseño electrónico, con la diferencia de que ahora será implementado en un software de simulación de circuitos electrónicos, en este caso bajo la plataforma Proteus[16], donde se puede apreciar el comportamiento de nuestros diseños, aunque suele ser algo ideal, está muy cercano a la realidad.

Comenzando por el bloque de procesamiento de audio, se definirá el diseño de los filtros con los parámetros deseados.

Diseño del filtro pasa bajas

En el diseño del sistema se estableció una etapa de filtro pasa bajas con una frecuencia de corte ω_C de 250Hz. Así pues, para un condensador de 0.01µf, se obtiene un valor de resistencia de 63.661kΩ. Para un factor de calidad de 0.707, la ganancia de la red de amplificación no inversora debe ser de 1,585 V/V, lo cual se logra haciendo RA=10kΩ y RB=5.8kΩ. Para normalizar la ganancia del filtro pasante es necesario construir una red atenuadora posterior al filtro, con una ganancia 1/µ, es decir, 0.630V/V. El circuito final se presenta en la imagen 6.21 y su grafica de respuesta en frecuencia en la imagen 6.22.

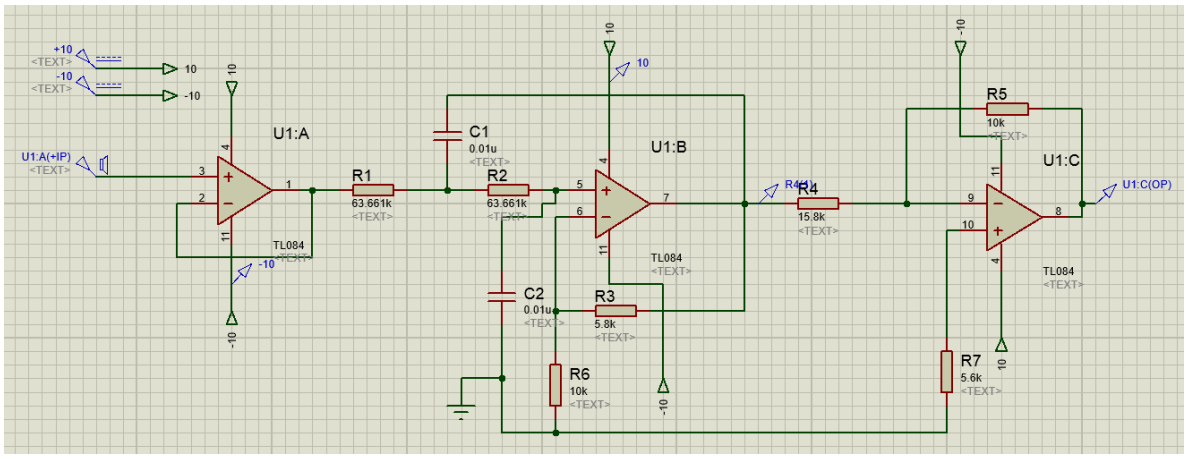


Imagen 6.21. Diseño del filtro pasa bajas con frecuencia de corte de 250Hz.

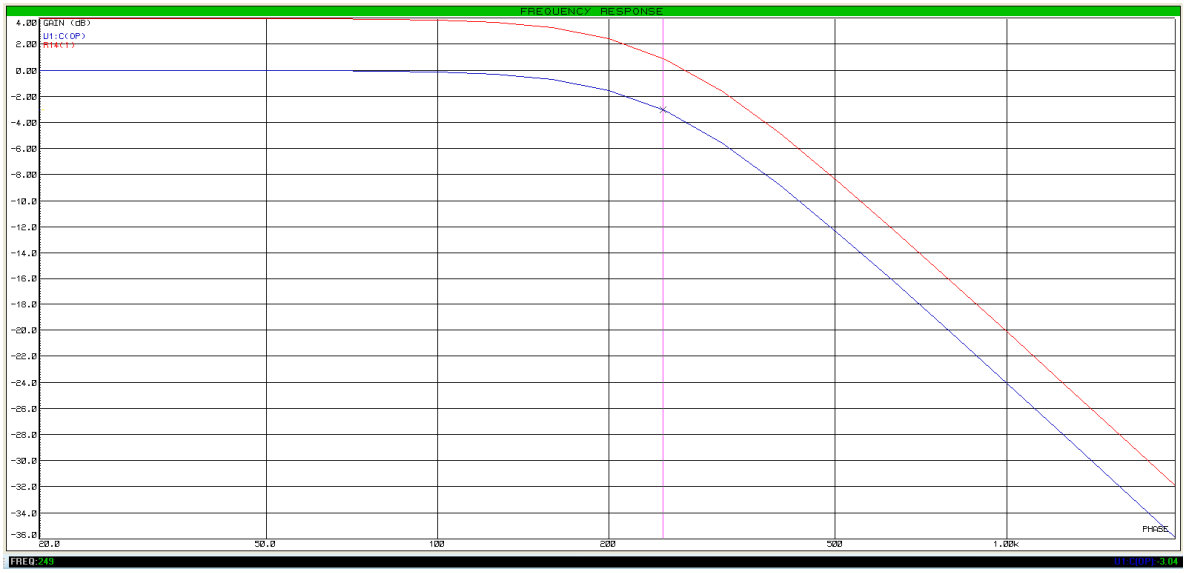


Imagen 6.22. Grafica Bode del filtro pasa bajos

Diseño del filtro pasa altas

Para el proyecto se diseñó una red de filtro pasa altas con una frecuencia de corte ω_C de 4 kHz. Según lo anterior, y empleando condensadores de 0.01µf, las resistencias a utilizar son de 3.979 kΩ. Como la ecuación para calcular la ganancia de la etapa de amplificación no inversora es idéntica a la del filtro pasa bajas, los circuitos de ajuste de factor de calidad y normalización de la ganancia en la banda pasante, son idénticos a los del pasa bajas. La imagen 6.23 presenta el circuito diseñado para esta etapa.

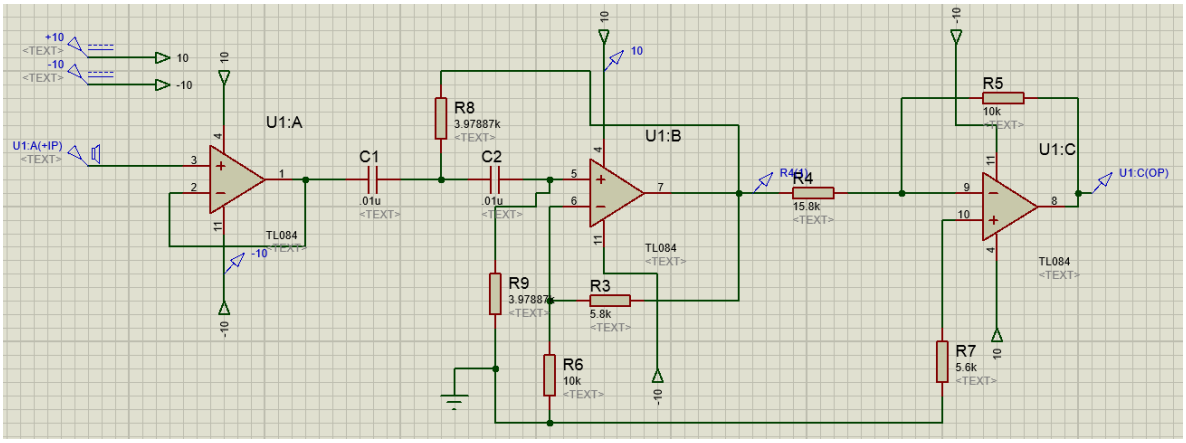


Imagen 6.23. Diseño del filtro pasa altos con frecuencia de corte de 4kHz.

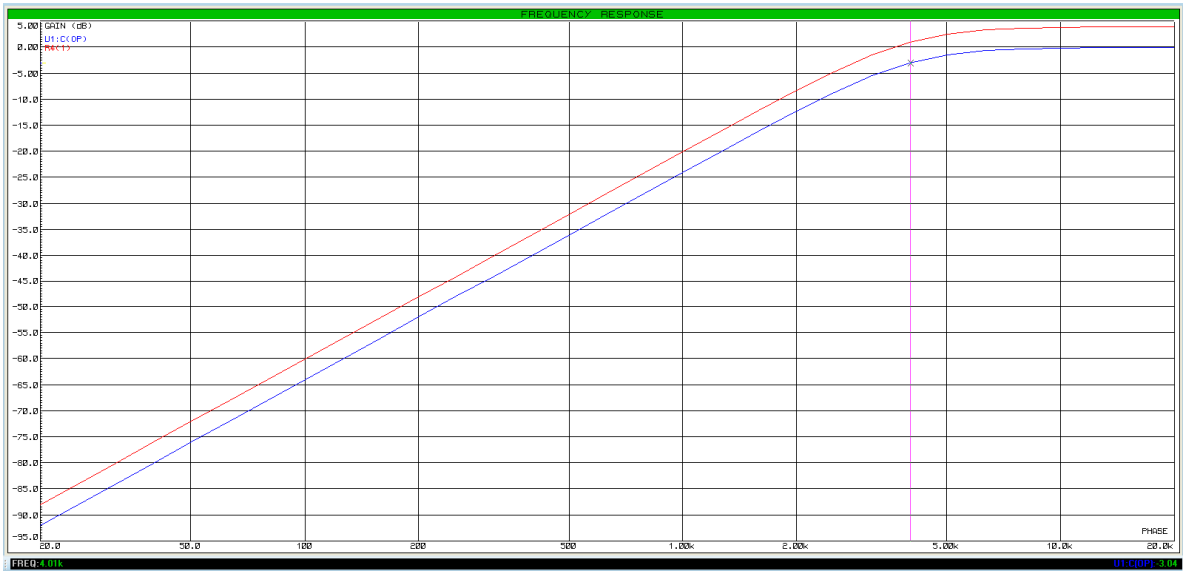


Imagen 6.24. Grafica Bode del filtro pasa altas

Diseño del filtro pasa banda

Para obtener un sistema que permita el paso de los componentes espectrales pertenecientes al intervalo $\omega_L - \omega_H$, con una atenuación inferior a 3dB, es posible conectar un filtro pasa bajas en cascada con un pasa altas, con frecuencias de corte configuradas en ω_H y ω_L respectivamente. La etapa presentada contiene un ancho de banda entre 500Hz y 2kHz, y fue diseñado siguiendo los criterios antes establecidos. Este circuito se presenta en la imagen 6.25 con su respectiva respuesta en frecuencia en la imagen 6.26.

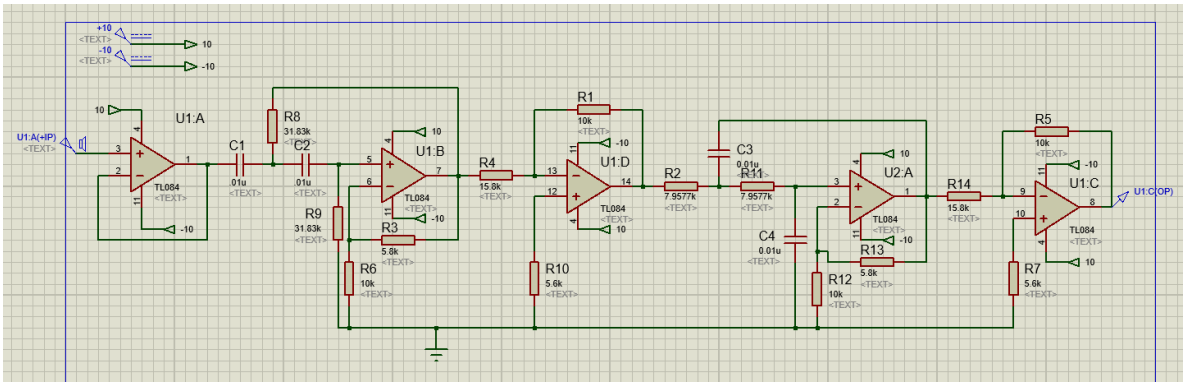


Imagen 6.23. Filtro pasa banda, con un ancho de banda de 500Hz a 2kHz.

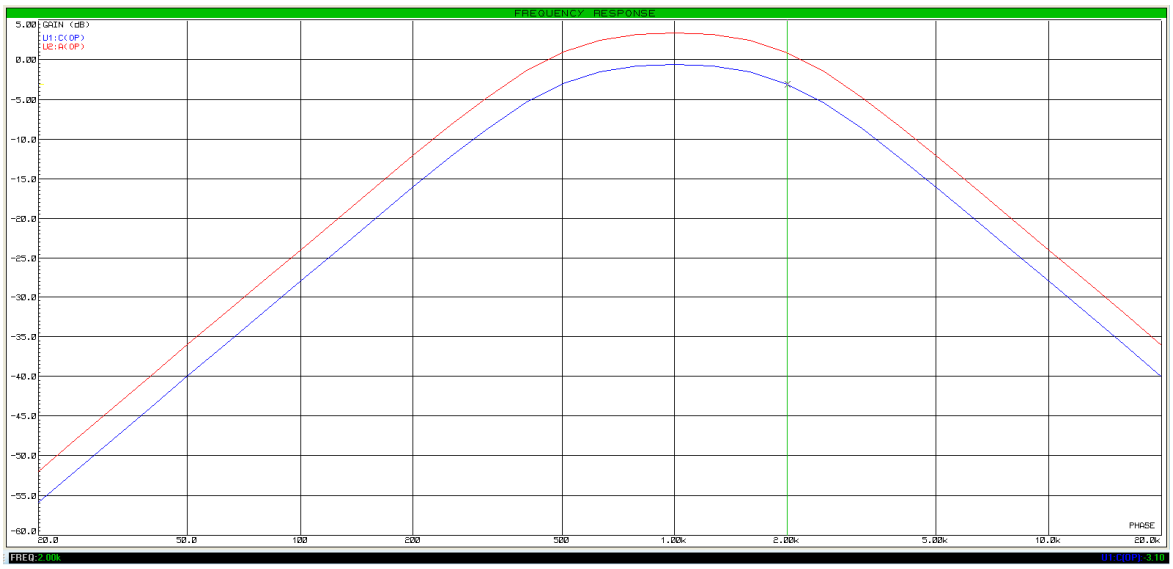


Imagen 6.24. Grafica Bode del filtro pasa banda

Posteriormente se realizó el diseño de la etapa de rectificación y promediado (imagen 6.25) de la señal de salida de cada uno de los filtros (un Rectificador-Promediador por filtro) para cada caso el valor de la resistencia R1 varía para ajustarse a la frecuencia de la señal, para tal efecto se implementaron los circuitos en una Protoboard, utilizando el circuito integrado TL074 para los filtros por ser un OPAMP de bajo ruido, dicho circuito se presenta en la imagen 6.26.

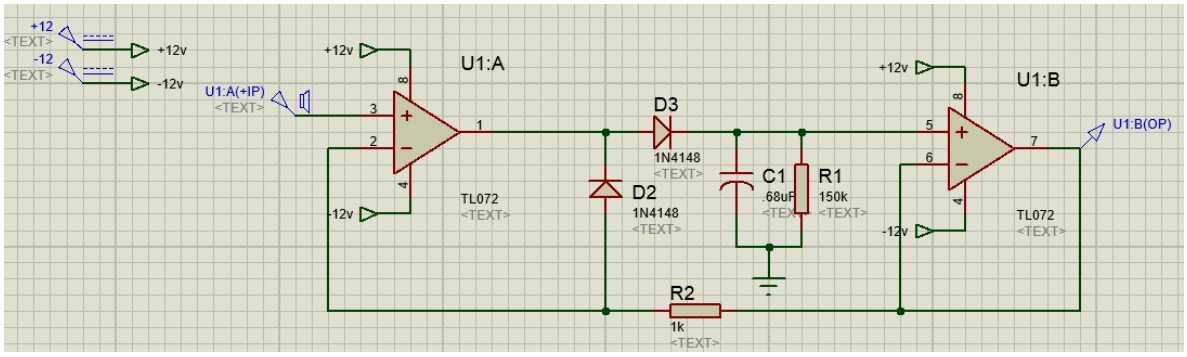


Imagen 6.25. Diseño del Rectificador-Promediador.

En el caso de las señales del filtro pasa bajas, el valor designado para R1 experimentalmente con ayuda de un potenciómetro es de 46.4kΩ, 71.5kΩ para el caso del filtro pasa banda y 37.4kΩ para el filtro pasa altas, valores de resistencia existente comercialmente en la gama de los resistores de precisión.

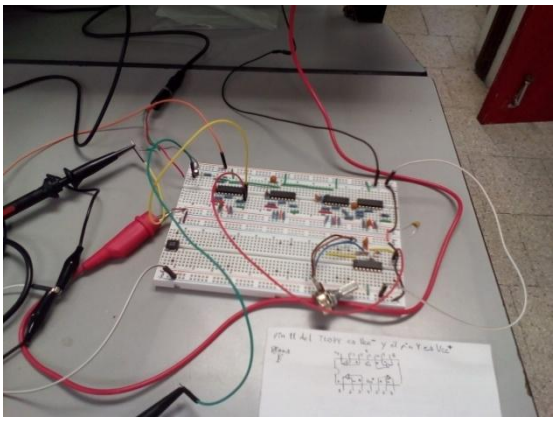


Imagen 6.26. Comprobación del funcionamiento del diseño en protoboard.

Por último se presenta una parte del circuito utilizado para generar las señales lógicas que recibirá el microcontrolador, implementado con comparadores de voltaje LM311, divisores de voltaje y una resistencia como acoplador de impedancia. (Imagen 6.27)

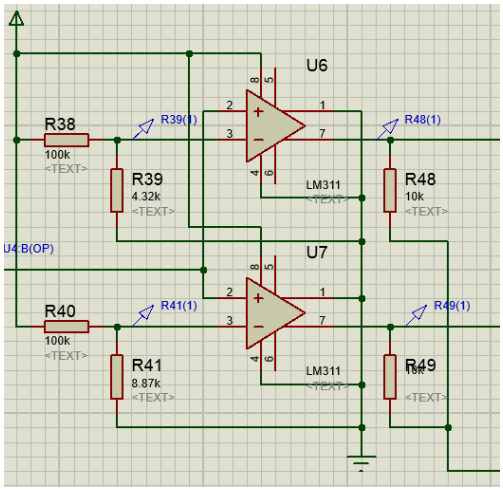
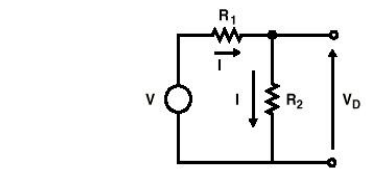


Imagen 6.27. Comparadores de voltaje.

Para definir los valores de los resistores en los 8 divisores de voltaje del sistema, se propuso un valor de 100kΩ para el primer resistor, posteriormente se despejo R2 de la ecuación (3), quedando como $R_2 = \frac{\left(\frac{V_D}{V}\right)R_1}{1-\left(\frac{V_D}{V}\right)}$ (4)

Se definieron 5 niveles de referencia básicos para los cuales se obtuvieron los siguientes valores resistivos utilizando la ecuación (4).



$V = IR_1 + IR_2 = I(R_1 + R_2)$ (1)

$I = \frac{V}{(R_1 + R_2)}$ (2)

$V_D = IR_2 = \frac{V}{(R_1 + R_2)}(R_2) = V\frac{R_2}{R_1 + R_2}$ (3)

Imagen 6.27.1

Nivel 1: $\mathrm{R}_2 = 2.1k\Omega$ para un $\mathrm{V}_\mathrm{D} = 200\mathrm{mV}$, $V = 5\mathrm{V}$ y $R_1 = 100\mathrm{k}\Omega$

Nivel 2: $\mathrm{R}_2 = 7.68k\Omega$ para un $\mathrm{V}_\mathrm{D} = 350\mathrm{mV}$, $V = 5\mathrm{V}$ y $R_1 = 100\mathrm{k}\Omega$

Nivel 3: $\mathrm{R}_2 = 8.87k\Omega$ para un $\mathrm{V}_\mathrm{D} = 400\mathrm{mV}$, $V = 5\mathrm{V}$ y $R_1 = 100\mathrm{k}\Omega$

Nivel 4: $\mathrm{R}_2 = 12.4k\Omega$ para un $\mathrm{V}_\mathrm{D} = 550\mathrm{mV}$, $V = 5\mathrm{V}$ y $R_1 = 100\mathrm{k}\Omega$

Nivel 5: $\mathrm{R}_2 = 23.7k\Omega$ para un $\mathrm{V}_\mathrm{D} = 950\mathrm{mV}$, $V = 5\mathrm{V}$ y $R_1 = 100\mathrm{k}\Omega$

Para el diseño del sistema de procesamiento maestro, se eligió el microcontrolador AVR "ATMEGA32" [17] por su capacidad de procesamiento y su amplia memoria de 32Kbytes. En la imagen 6.28 se observa el diseño de la placa principal, con sus conexiones, puertos de entrada y salida, configuración manual, y de comunicación Serial y SPI.

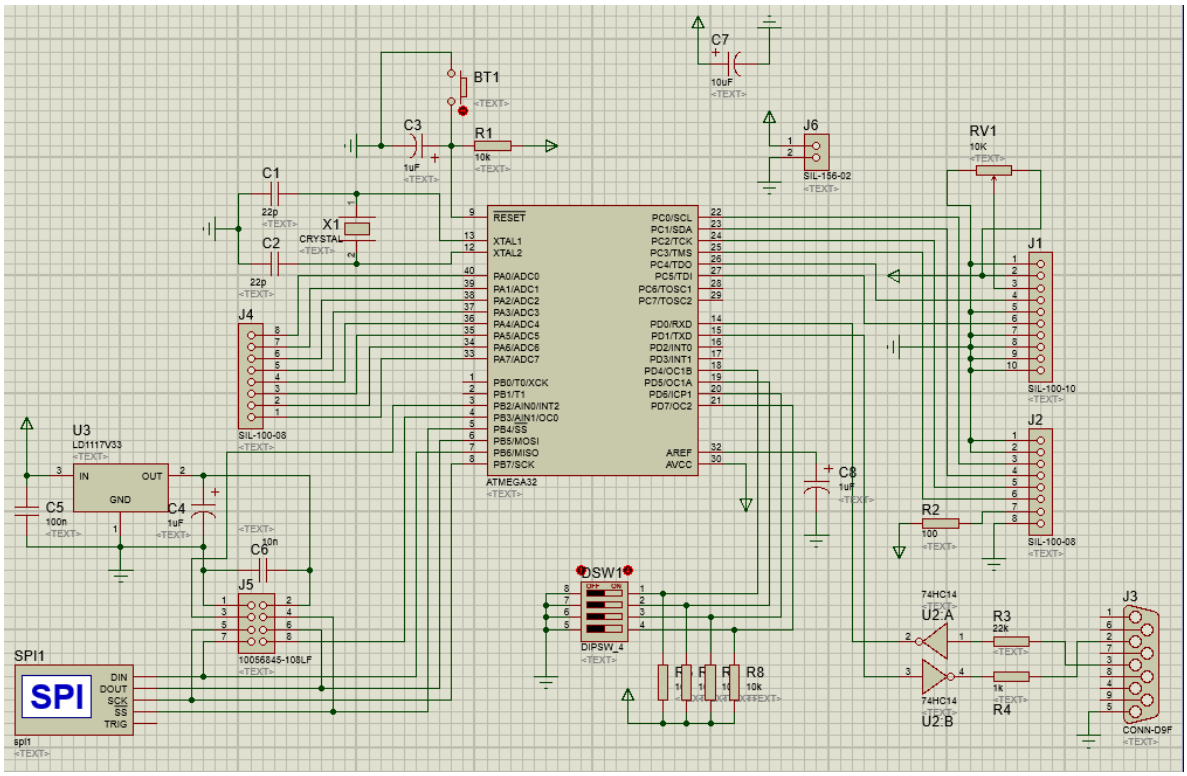


Imagen 6.28. Diseño de la placa principal del sistema maestro.

Los diseños electrónicos antes presentados corresponden al sistema maestro y a continuación se detallan los diseños correspondientes al sistema esclavo.

Para el sistema esclavo se seleccionó el microcontrolador AVR "ATMEGA8535" [18] ya que no requiere de tantos recursos de procesamiento de datos y almacenamiento en memoria ya que este AVR solo cuenta con 8Kbytes.

En la imagen 6.29, se muestra el diseño de las conexiones correspondientes a los distintos dispositivos conectados de forma modular, diseño útil para poder realizar cambios en caso fallas, lo cual facilita su reemplazo.

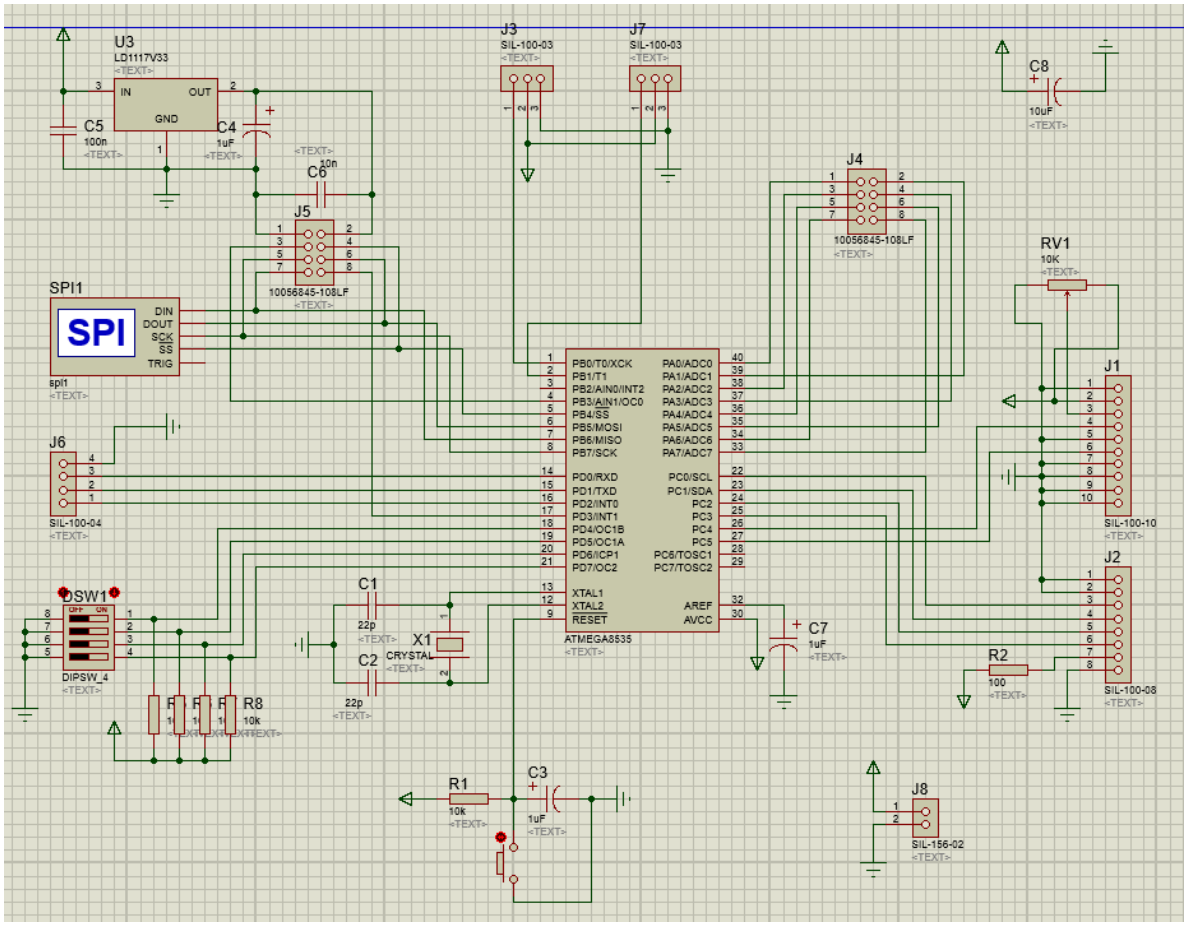


Imagen 6.29. Diseño de la placa principal del sistema esclavo.

Al considerar un diseño modular se excluyen varios circuitos que conforman el sistema esclavo dentro de la placa principal, los demás módulos se diseñan por separado y se considera la interconexión entre ellos a través de cableado y conectores.

Puesto que los motores trabajan a niveles voltaje y corriente distintos que un microcontrolador, se diseñó un driver con diodos y puentes H L298, con interfaz a uno de los puertos del microcontrolador pero con conexiones por separado a la fuente de voltaje, a continuación podemos apreciar el diseño de este driver en la imagen 6.30.

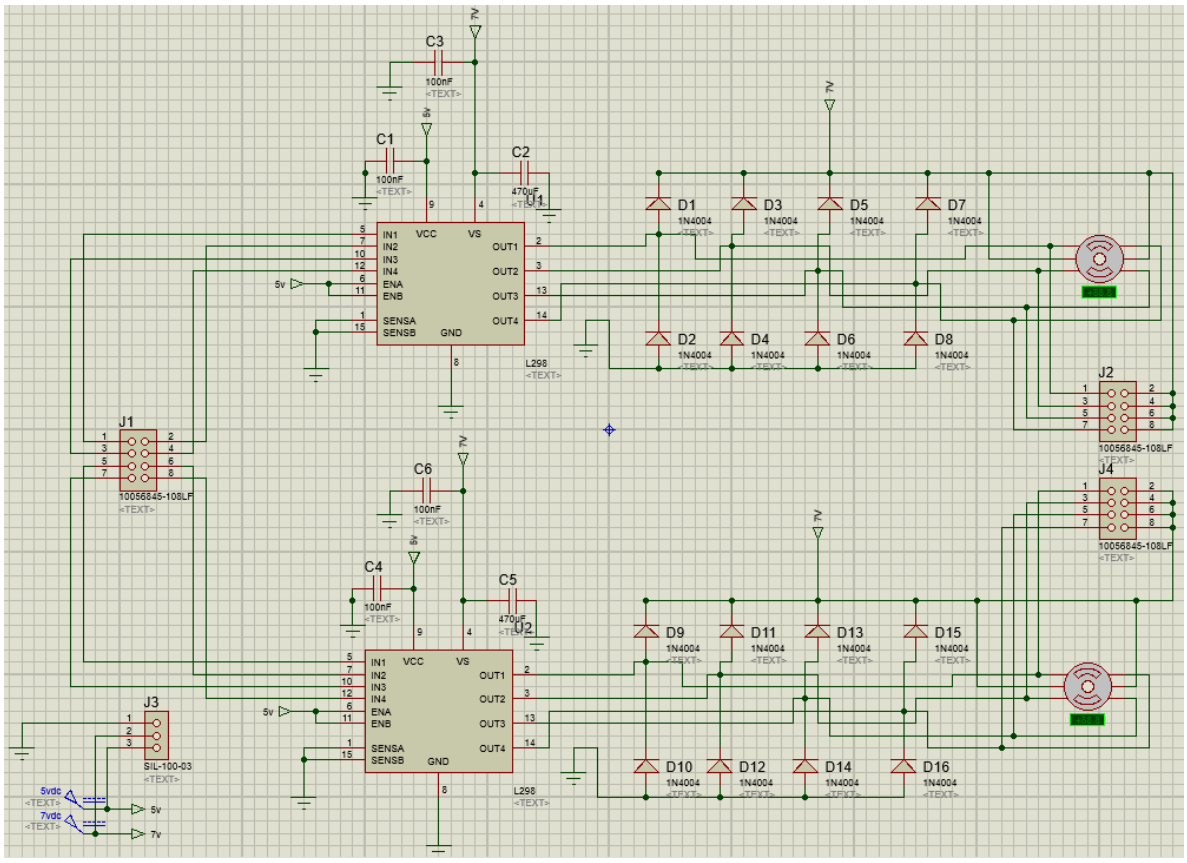


Imagen 6.30. Diseño del driver para los motores a pasos.

Otro de los módulos está conformado por el driver de los LEDs de potencia, que de igual manera no pueden ser conectados directamente a los pines del microcontrolador por las diferencias en la demanda de voltaje y corriente que son mayores a las proporcionadas por el antes mencionado. (Imagen 6.31)

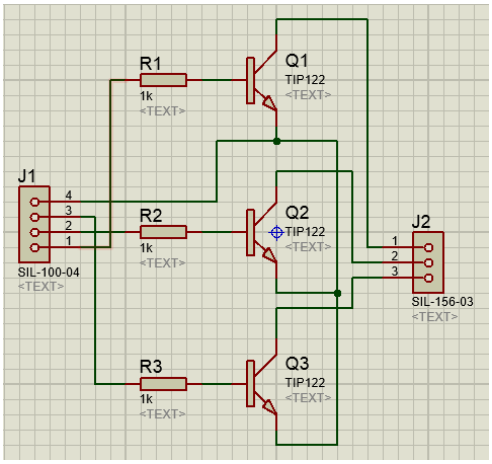


Imagen 6.31. Driver para el encendido de los LEDs.

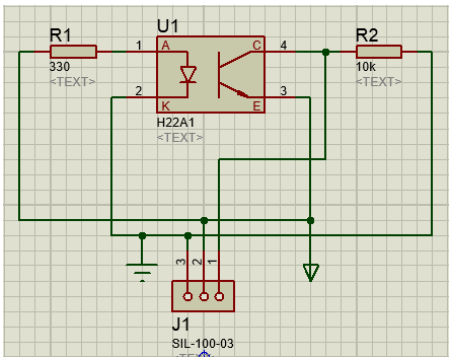


Imagen 6.32. Sensor del sistema encoder

El diseño mostrado en la imagen 3.32, trata de la implementación de un optoaislador para ser utilizado como un sistema encoder que sirve para controlar el movimiento de los motores.

Como último paso a seguir, una vez que fueron corregidos los errores y se verificó el correcto funcionamiento con la implementación de de los circuitos en la protoboard, se procedió a realizar el diseño de las placas de circuito impreso (PCB) los cuales se encuentran en el "Anexo B".

6.3. Desarrollo de drivers y software

Para el desarrollo de esta etapa se consideraron toras las tareas que el prototipo sería capaz de efectuar, movimientos, colores de la luz, comunicación serial e inalámbrica, visualización de estados, entre otros.

Los drivers de los sistemas maestro y esclavo se desarrollaron en la plataforma desarrollada por Atmel, conocida como Atmel Studio, empresa a la cual pertenecen los microcontrolades utilizados en los sistemas mencionados al principio de este párrafo.

USART AVR programacion en el ATMEL STUDIO

Configuración del registro UCSR0A. Todos los bits de este registro se pondrán a 0 ya que a excepción del bit bit1 U2X0 y el bit0 todos los demás bits trabajan en forma automática, el bit1 se pone a 0 porque se trabajará en la velocidad de los baudios en forma normal, y el bit 0 se pone a 0 porque se usará el modo asíncrono.

Luego en la inicialización del módulo USART AVR el registro UCSR0A en el ATMEL STUDIO quedará así:

UCSR0A=0b00000000;// velocidad normal

Configuración del registro UCSR0B no se utilizarán interrupciones, se habilitarán los pines RXD y TXD para la comunicación serial y la comunicación será a 8 bits
El bit7 se pondrá a 0 porque no usará la interrupción USART AVR por recepción.
El bit6 se pondrá a 0 porque no usará la interrupción USART AVR por transmisión.
El bit5 se pondrá a 0 porque no usará la interrupción USART AVR la detección de que el registro UDR0 se quedó vacío.

El bit4 se pondrá este bit a 1 para habilitar el uso del pin RXD en la recepción del módulo USART AVR. Se habilita es uso de la recepción.

El bit3 se pondrá este bit a 1 para habilitar el uso del pin TXD para la transmisión del módulo USART AVR. Se habilita es uso de la transmisión.

El bit2 Este bit junto con los bits 2 y 1 del registro UCSR0C son para elegir de cuantos bits serán los datos a recibir o transmitir en la comunicación serial, se utilizará la comunicación serial a 8 bits, por lo que este bit se pondrá a 0.

El bit1 se pone a 0 ya que este bit será el noveno bit en la recepción del dato, si la comunicación es a 9 bits.

El bit0 se pone a 0 ya que este bit será el noveno bit en la transmisión del dato, si la comunicación es a 9 bits.

Luego en la inicialización del módulo USART AVR el registro UCSR0B en el ATMEL STUDIO quedará así:

UCSR0B=0b00011000;//transmisión y recepción habilitados a 8 bits

Configuración del registro UCSR0C se elegirá el modo asíncrono, sin bit de paridad, un bit de parada.

Los bits 7 y 6 se ponen a 0 para elegir el modo asíncrono del módulo USART AVR.
Los bits 5 y 4 se ponen a 0 para no utilizar bit de paridad.
El bit3 se pone a 0 para elegir un solo bit de parada.
Los bits 2 y 1 se ponen a 1 para elegir que la comunicación serial será a 8 bits.
El bit0 Este bit es utilizado en el bit síncrono, en el modo asíncrono se pondrá a 0.

Luego en la inicialización del módulo USART AVR el registro UCSR0B en el ATMEL STUDIO quedará así:

UCSR0C=0b00000110;//asíncrono, sin bit de paridad, 1 bit de parada a 8 bits

Carga del registro UBRR0 para los baudios o la cantidad de bits por segundo para la comunicación serial, como será a velocidad normal por lo que el bit U2X0 del registro UCSR0A se puso a 0 y en forma asíncrona, se utilizará la fórmula.

Velocidad en baudios=baudios=Fosc/(16*(UBRR0+1))

Al despejar se tendrá

UBRR0=(Fosc/(16*baudios))-1

En los ejemplos que se harán se utilizará una Fosc de 8Mhz y la velocidad será de 9600 baudios, por lo que al reemplazar se tendrá que el valor a cargar en el registro UBRR0 será de 51, luego en el ATMEL STUDIO será:

UBRR0=51;//para una velocidad de 9600 baudios con un oscilador de 8Mhz

Entonces se puede iniciar el módulo USART AVR en el ATMEL STDIO de la siguiente manera:

```
//inicialización del módulo USART AVR modo asíncrono, a 8bits,
//a 9600 baudios, sin bit de paridad, 1 bit de parada
UCSR0A=0b00000000; // velocidad normal
UCSR0B=0b00011000; //transmisión y recepción habilitados a 8 bits
UCSR0C=0b00000110; //asíncrono, sin bit de paridad, 1 bit de parada a 8 bits
UBRR0=51; //para una velocidad de 9600 baudios con un oscilador de 8Mhz
```

La inicialización podría hacerse dentro de una función a la que se le puede dar el nombre que se quiera, en este caso se la llamará iniciar_usart(), además el registro UCSR0A siempre se inicia a 0 por lo que no es necesario inicializarlo en este caso ya que se necesita que sea 0, lo que en el ATMEL STUDIO será así:

```
////////////////////////////////////////////
//inicialización del módulo USART AVR modo asíncrono
//en una función, a 8bits,a 9600 baudios, sin bit de paridad
//1 bit de parada
////////////////////////////////////////////////
void iniciar_usart(){
UCSR0B=0b00011000; //transmisión y recepción habilitados a 8 bits
UCSR0C=0b00000110; //asíncrono, sin bit de paridad, 1 bit de parada a 8 bits
UBRR0=51; //para una velocidad de 9600 baudios con un oscilador de 8Mhz
}
```

Para la recepción de datos con el módulo USART AVR.

Los datos que se reciben son de tipo caracter, cuando se va recibir un dato a través del pin RXD se tendrá que esperar a que el bit7 del registro UCSR0A se ponga a 1, este bit indica que se ha completado la recepción del dato, el dato recibido es de tipo caracter char y estará en el registro UDR0, cuando se lee el dato guardándolo en alguna variable de tipo caracter char este bit se pondrá automáticamente a 0, la recepción de los datos en el ATMEL STUDIO se hará en una función a la que se le puede llamar como se desee, en este caso se le llamará recibe_caracter_usart().

```
////////////////////////////////////////////////
//recepción de datos del módulo USART AVR modo asíncrono
////////////////////////////////////////////////////////
 unsigned char recibe_caracter_usart(){
if(UCSR0A&(1<<7)){  //si el bit7 del registro UCSR0A se ha puesto a 1
return UDR0;    //devuelve el dato almacenado en el registro UDR0
}
else//sino
return 0;//retorna 0
}
```

Para la transmisión de datos con el módulo USART AVR.

Los datos que se transmiten son de tipo caracter, cuando se va transmitir un dato a través del pin TXD se tendrá que esperar a que el registro UDR0 donde se ponen los datos que se van a enviar esté vacío, esto se hace esperando a que el bit5 UDRE0 del registro UCSR0A se ponga a 1, este bit indica que se ha completado la transmisión de un dato, el dato transmitido es de tipo caracter char, cuando se vuelve a cargar otro dato en el registro UDR0 este bit se pondrá automáticamente a 0, la transmisión de los datos en el ATMEL STUDIO se hará en una función a la que se le puede llamar como se desee, en este caso se le llamará envia_caracter_usart().

```
////////////////////////////////////////////////
//transmisión de datos del módulo USART AVR modo asíncrono
////////////////////////////////////////////////
 void envia_caracter_usart(unsigned char caracter){
while(!(UCSR0A&(1<<5)));    // mientras el registro UDR0 esté lleno espera
UDR0 = caracter;            //cuando el el registro UDR0 está vacio se envia el caracter
}
```

Para la transmisión de cadenas de caracteres con el módulo USART AVR se puede utilizar la función vista anteriormente para la transmisión de caracteres para enviar cadenas de caracteres, para ello se creará otra función que se puede llamar como se desee pero en este caso se le llamará envia_cadena_usart().

```
//////////////////////////////////////////////
//transmisión de cadenas de caracteres con el módulo USART AVR modo asíncrono
//////////////////////////////////////////////
 void envia_cadena_usart(char* cadena){ //cadena de caracteres de tipo char
while(*cadena !=0x00){ //mientras el último valor de la cadena sea diferente
//a el caracter nulo
envia_caracter_usart(*cadena); //transmite los caracteres de cadena
cadena++; //incrementa la ubicación de los caracteres en cadena
//para enviar el siguiente caracter de cadena
}
}
```

Uso del display LCD 16x2

```
//líneas de cabecera para el uso de la lcd
#define D4 eS_PORTC0
#define D5 eS_PORTC1
#define D6 eS_PORTC2
#define D7 eS_PORTC3
#define RS eS_PORTC4
#define EN eS_PORTC5

#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include "lcd.h"


//archivo lcd. h
#define eS_PORTA0 0
#define eS_PORTA1 1
#define eS_PORTA2 2
#define eS_PORTA3 3
#define eS_PORTA4 4
#define eS_PORTA5 5
#define eS_PORTA6 6
#define eS_PORTA7 7
#define eS_PORTB0 10
#define eS_PORTB1 11
#define eS_PORTB2 12
#define eS_PORTB3 13
#define eS_PORTB4 14
```

```
#define eS_PORTB5 15
#define eS_PORTB6 16
#define eS_PORTB7 17
#define eS_PORTC0 20
#define eS_PORTC1 21
#define eS_PORTC2 22
#define eS_PORTC3 23
#define eS_PORTC4 24
#define eS_PORTC5 25
#define eS_PORTC6 26
#define eS_PORTC7 27
#define eS_PORTD0 30
#define eS_PORTD1 31
#define eS_PORTD2 32
#define eS_PORTD3 33
#define eS_PORTD4 34
#define eS_PORTD5 35
#define eS_PORTD6 36
#define eS_PORTD7 37

#ifndef D0
#define D0 eS_PORTA0
#define D1 eS_PORTA1
#define D2 eS_PORTA2
#define D3 eS_PORTA3
#endif

#include<util/delay.h>

void pinChange(int a, int b)
{
        if(b == 0)
        {
                if(a == eS_PORTA0)
                  PORTA &= ~(1<<PA0);
                else if(a == eS_PORTA1)
                  PORTA &= ~(1<<PA1);
                else if(a == eS_PORTA2)
                  PORTA &= ~(1<<PA2);
                else if(a == eS_PORTA3)
                  PORTA &= ~(1<<PA3);
                else if(a == eS_PORTA4)
                  PORTA &= ~(1<<PA4);
                else if(a == eS_PORTA5)
                  PORTA &= ~(1<<PA5);
                else if(a == eS_PORTA6)
                  PORTA &= ~(1<<PA6);
                else if(a == eS_PORTA7)
                  PORTA &= ~(1<<PA7);
                else if(a == eS_PORTB0)
                  PORTB &= ~(1<<PB0);
```

```
            else if(a == eS_PORTB1)
              PORTB &= ~(1<<PB1);
            else if(a == eS_PORTB2)
              PORTB &= ~(1<<PB2);
            else if(a == eS_PORTB3)
              PORTB &= ~(1<<PB3);
            else if(a == eS_PORTB4)
              PORTB &= ~(1<<PB4);
            else if(a == eS_PORTB5)
              PORTB &= ~(1<<PB5);
            else if(a == eS_PORTB6)
              PORTB &= ~(1<<PB6);
            else if(a == eS_PORTB7)
              PORTB &= ~(1<<PB7);
            else if(a == eS_PORTC0)
              PORTC &= ~(1<<PC0);
            else if(a == eS_PORTC1)
              PORTC &= ~(1<<PC1);
            else if(a == eS_PORTC2)
              PORTC &= ~(1<<PC2);
            else if(a == eS_PORTC3)
              PORTC &= ~(1<<PC3);
            else if(a == eS_PORTC4)
              PORTC &= ~(1<<PC4);
            else if(a == eS_PORTC5)
              PORTC &= ~(1<<PC5);
else if(a == eS_PORTC6)
  PORTC &= ~(1<<PC6);
            else if(a == eS_PORTC7)
              PORTC &= ~(1<<PC7);
            else if(a == eS_PORTD0)
              PORTD &= ~(1<<PD0);
            else if(a == eS_PORTD1)
              PORTD &= ~(1<<PD1);
            else if(a == eS_PORTD2)
              PORTD &= ~(1<<PD2);
            else if(a == eS_PORTD3)
              PORTD &= ~(1<<PD3);
            else if(a == eS_PORTD4)
              PORTD &= ~(1<<PD4);
            else if(a == eS_PORTD5)
              PORTD &= ~(1<<PD5);
            else if(a == eS_PORTD6)
              PORTD &= ~(1<<PD6);
            else if(a == eS_PORTD7)
              PORTD &= ~(1<<PD7);
    }
    else
    {
            if(a == eS_PORTA0)
              PORTA |= (1<<PA0);
```

```
else if(a == eS_PORTA1)
  PORTA |= (1<<PA1);
else if(a == eS_PORTA2)
  PORTA |= (1<<PA2);
else if(a == eS_PORTA3)
  PORTA |= (1<<PA3);
else if(a == eS_PORTA4)
  PORTA |= (1<<PA4);
else if(a == eS_PORTA5)
  PORTA |= (1<<PA5);
else if(a == eS_PORTA6)
  PORTA |= (1<<PA6);
else if(a == eS_PORTA7)
  PORTA |= (1<<PA7);
else if(a == eS_PORTB0)
  PORTB |= (1<<PB0);
else if(a == eS_PORTB1)
  PORTB |= (1<<PB1);
else if(a == eS_PORTB2)
  PORTB |= (1<<PB2);
else if(a == eS_PORTB3)
  PORTB |= (1<<PB3);
else if(a == eS_PORTB4)
  PORTB |= (1<<PB4);
else if(a == eS_PORTB5)
  PORTB |= (1<<PB5);
else if(a == eS_PORTB6)
  PORTB |= (1<<PB6);
else if(a == eS_PORTB7)
  PORTB |= (1<<PB7);
else if(a == eS_PORTC0)
  PORTC |= (1<<PC0);
else if(a == eS_PORTC1)
  PORTC |= (1<<PC1);
else if(a == eS_PORTC2)
  PORTC |= (1<<PC2);
else if(a == eS_PORTC3)
  PORTC |= (1<<PC3);
else if(a == eS_PORTC4)
  PORTC |= (1<<PC4);
else if(a == eS_PORTC5)
  PORTC |= (1<<PC5);
else if(a == eS_PORTC6)
  PORTC |= (1<<PC6);
else if(a == eS_PORTC7)
  PORTC |= (1<<PC7);
else if(a == eS_PORTD0)
  PORTD |= (1<<PD0);
else if(a == eS_PORTD1)
  PORTD |= (1<<PD1);
else if(a == eS_PORTD2)
```

```
                PORTD |= (1<<PD2);
            else if(a == eS_PORTD3)
                PORTD |= (1<<PD3);
            else if(a == eS_PORTD4)
                PORTD |= (1<<PD4);
            else if(a == eS_PORTD5)
                PORTD |= (1<<PD5);
            else if(a == eS_PORTD6)
                PORTD |= (1<<PD6);
            else if(a == eS_PORTD7)
                PORTD |= (1<<PD7);
        }
}

//LCD 8 Bit
void Lcd8_Port(char a)
{
        if(a & 1)
        pinChange(D0,1);
        else
        pinChange(D0,0);

        if(a & 2)
        pinChange(D1,1);
        else
        pinChange(D1,0);

        if(a & 4)
        pinChange(D2,1);
        else
        pinChange(D2,0);

        if(a & 8)
        pinChange(D3,1);
        else
        pinChange(D3,0);

        if(a & 16)
        pinChange(D4,1);
        else
        pinChange(D4,0);

        if(a & 32)
        pinChange(D5,1);
        else
        pinChange(D5,0);

        if(a & 64)
        pinChange(D6,1);
        else
        pinChange(D6,0);
```

```
        if(a & 128)
        pinChange(D7,1);
        else
        pinChange(D7,0);
}
void Lcd8_Cmd(char a)
{
        pinChange(RS,0);            // => RS = 0
        Lcd8_Port(a);             //Data transfer
        pinChange(EN,1);            // => E = 1
        _delay_ms(1);
        pinChange(EN,0);            // => E = 0
        _delay_ms(1);
}

void Lcd8_Clear()
{
        Lcd8_Cmd(1);
}

void Lcd8_Set_Cursor(char a, char b)
{
        if(a == 1)
        Lcd8_Cmd(0x80 + b);
        else if(a == 2)
        Lcd8_Cmd(0xC0 + b);
}

void Lcd8_Init()
{
        pinChange(RS,0);
        pinChange(EN,0);
        _delay_ms(20);
        //////////// Reset process from datasheet /////////
        Lcd8_Cmd(0x30);
        _delay_ms(5);
        Lcd8_Cmd(0x30);
        _delay_ms(1);
        Lcd8_Cmd(0x30);
        _delay_ms(10);
        ////////////////////////////////////////////////
        Lcd8_Cmd(0x38);    //function set
        Lcd8_Cmd(0x0C);    //display on,cursor off,blink off
        Lcd8_Cmd(0x01);    //clear display
        Lcd8_Cmd(0x06);    //entry mode, set increment
}

void Lcd8_Write_Char(char a)
{
        pinChange(RS,1);            // => RS = 1
```

```
        Lcd8_Port(a);           //Data transfer
        pinChange(EN,1);            // => E = 1
        _delay_ms(1);
        pinChange(EN,0);            // => E = 04
        _delay_ms(1);
}

void Lcd8_Write_String(char *a)
{
        int i;
        for(i=0;a[i]!='\0';i++)
        Lcd8_Write_Char(a[i]);
}

void Lcd8_Shift_Right()
{
        Lcd8_Cmd(0x1C);
}

void Lcd8_Shift_Left()
{
        Lcd8_Cmd(0x18);
}
//End LCD 8 Bit Interfacing Functions

//LCD 4 Bit Interfacing Functions

void Lcd4_Port(char a)
{
        if(a & 1)
        pinChange(D4,1);
        else
        pinChange(D4,0);

        if(a & 2)
        pinChange(D5,1);
        else
        pinChange(D5,0);

        if(a & 4)
        pinChange(D6,1);
        else
        pinChange(D6,0);

        if(a & 8)
        pinChange(D7,1);
        else
        pinChange(D7,0);
}
void Lcd4_Cmd(char a)
{
```

```
        pinChange(RS,0);            // => RS = 0
        Lcd4_Port(a);
        pinChange(EN,1);           // => E = 1
        _delay_ms(1);
        pinChange(EN,0);            // => E = 0
        _delay_ms(1);
}

void Lcd4_Clear()
{
        Lcd4_Cmd(0);
        Lcd4_Cmd(1);
}

void Lcd4_Set_Cursor(char a, char b)
{
        char temp,z,y;
        if(a == 1)
        {
                temp = 0x80 + b;
                z = temp>>4;
                y = (0x80+b) & 0x0F;
                Lcd4_Cmd(z);
                Lcd4_Cmd(y);
        }
        else if(a == 2)
        {
                temp = 0xC0 + b;
                z = temp>>4;
                y = (0xC0+b) & 0x0F;
                Lcd4_Cmd(z);
                Lcd4_Cmd(y);
        }
}

void Lcd4_Init()
{
        Lcd4_Port(0x00);
        _delay_ms(20);
        //////////// Reset process from datasheet /////////
        Lcd4_Cmd(0x03);
        _delay_ms(5);
        Lcd4_Cmd(0x03);
        _delay_ms(11);
        Lcd4_Cmd(0x03);
        ////////////////////////////////////////////////
        Lcd4_Cmd(0x02);
        Lcd4_Cmd(0x02);
        Lcd4_Cmd(0x08);
        Lcd4_Cmd(0x00);
        Lcd4_Cmd(0x0C);
```

```
        Lcd4_Cmd(0x00);
        Lcd4_Cmd(0x06);
}

void Lcd4_Write_Char(char a)
{
        char temp,y;
        temp = a&0x0F;
        y = a&0xF0;
        pinChange(RS,1);            // => RS = 1
        Lcd4_Port(y>>4);            //Data transfer
        pinChange(EN,1);
        _delay_ms(1);
        pinChange(EN,0);
        _delay_ms(1);
        Lcd4_Port(temp);
        pinChange(EN,1);
        _delay_ms(1);
        pinChange(EN,0);
        _delay_ms(1);
}

void Lcd4_Write_String(char *a)
{
        int i;
        for(i=0;a[i]!='\0';i++)
        Lcd4_Write_Char(a[i]);
}

void Lcd4_Shift_Right()
{
        Lcd4_Cmd(0x01);
        Lcd4_Cmd(0x0C);
}

void Lcd4_Shift_Left()
{
        Lcd4_Cmd(0x01);
        Lcd4_Cmd(0x08);
}
```

```
/*****************************************************************************
**************
* usart.h
* Descripción:
*
* Libreria para el manejo de la USART de los microcontroladores
* AVR ATmega16 y ATmega32 escrita en el compilador GNU avr-gcc.
*
* La libreria implementa funcoines para inicializar la USART, enviar
* y recibir datos por el puerto serie,verificar si hay dato disponible
* y enviar cadenas de texto.
* Cuenta con la posibilidad hacer uso de las interrupciones para el
* envío y recepcion de datos, con lo que los datos recibidos y enviados
* se almacenan en un buffer temporarl (FIFO) que serán llenados o
* vaciados por las funciones de las interruciones por dato recibido o
* por transmisor vacío. Para hacer uso de las interrupciones se usan
* las definiciones:
* #define USE_USART_RX_INTETTUPT // Usa la interrupción de dato disponible en RX
* #define USE_USART_TX_INTERRUPT // Usa la interrupcion de transmisor vacío
* #define USE_USART_INTERRUPTS // Usa las 2 interrupciónes
*
* Uso:
* #define USE_USART_RX_INTERRUPT
* #include "usart.h"
*
* El programador puede activar y atender las interrupcions por su cuenta
* omitiendo las definiciones anteriormente mencionadas.
* Para el caso de que se usen interrupciones, se debe tomar en cuenta el
* tamaño de los buffer de entrada y salida (FIFO), ya que es espacio en
```

```
* memoria RAM y hay que considerar ese espacio en el desarrollo de la
* aplicacion. Por default se define un buffer de 64 bytes, que bien
* puede ser ampliado por el programador (a un maximo de 255 para los
* indices de tipo char) y que es util en microcontroladores de mayores
* capacidades.

* Se puede hacer uso de la entrada y salida estandar para el envio y
* recepcion de datos con la libreria "stdio.h", usando las funciones
* printf, scanf, putc, etc. Para el uso de esta propiedad hay que hacer
* la definicion "#define USE_USART_STDIO".
*
* Uso:
* #define USE_USART_STDIO
* #include "usart.h"
*
* Las funciones de la libreria son:

* ------------------------------------------------------------------------
* | void USART_Init(int baudrate) *
* | Inicializa la comunicación estandar (8N1) con el baudaje especificado.
* ------------------------------------------------------------------------
* | void USART_Init2(int baudrate,
* | unsigned char stop_bits,
* | unsigned char parity,
* | unsigned char data_size)
* |
* | Inicializa la comunicacion configurando todos los parametros.
* ------------------------------------------------------------------------
* | void USART_PutChar(char c)
* | Envía un dato por la USART o lo inserta en el buffer de salida si se
| usa interrupción.
```

```
* ------------------------------------------------------------------------
* | int USART_GetChar()
* | Lee un dato recibido en la USART.
* ------------------------------------------------------------------------
* | void USART_PutString(char *str)
* | Envia una cadena de caracteres por la USART.
* ------------------------------------------------------------------------
* | int USART_Kbhit()
* | Determina si hay un dato para ser leido de la USAR, o el numero
* | de datos en el buffer de entrada si se usa interrupcion.
* ------------------------------------------------------------------------
*
* Ejemplo de uso:
* #define F_CPU 8000000L
* #define USE_USART_RX_INTERRUPT
* #define USE_USART_TX_INTERRUPT
* #define USE_USART_STDIO
*
* #include <avr/io.h>
* #include <util/delay.h>
* #include "usart.h"
*
* int main(void)
* {
* USART_Init(9600);
* sei();
*
* USART_PutString("Ejemplo ECO con la USART.\n");
*
* while(1){
```

```
*
* if(USART_Kbhit()){
*
* printf( "Tecla: %c" , USART_GetChar() );
* _delay_ms(10);
*
* }
* }
* }
**********/

#ifndef USART_H_
#define USART_H_

#include <avr/io.h>

#define USART_DATA_SIZE_5 0x00 /*Tamaño de palabra de 5 bits*/
#define USART_DATA_SIZE_6 _BV(UCSZ0) /*Tamaño de palabra de 6 bits*/
#define USART_DATA_SIZE_7 _BV(UCSZ1) /*Tamaño de palabra de 7 bits*/
#define USART_DATA_SIZE_8 _BV(UCSZ0) | _BV(UCSZ1) /*tamaño de palabra de 8 bits*/

#define USART_PARITY_OFF 0x00 /*Sin paridad*/
#define USART_PARITY_EVEN_ON _BV(UPM1) /*Paridad Par*/
#define USART_PARITY_ODD_ON _BV(UPM1) | _BV(UPM0) /*Paridad Impar*/

#define USART_STOP_BITS_1 0x00 /*1 bit de parada*/
#define USART_STOP_BITS_2 _BV(USBS) /*2 bit de parada*/

/**********************************************************************
* Macro calcular el valor de los registros generadores del
*baudrate segun la fórmula del datasheet.
**********************************************************************/
```

```
#define MYUBRR(BAUD) (F_CPU/(BAUD*8L)-1)/2 //Mejor formula que
(F_CPU/16/(BAUD)-1)

/**************************************************************************
* Declaración de Funciones
**************************************************************************/

void USART_Init(long baudrate);
void USART_Init2(long baudrate,unsigned char stop_bits, unsigned char parity,unsigned
char data_size);
void USART_PutChar(char c);
int USART_GetChar();
void USART_PutString(char *str);
int USART_Kbhit();

/**************************************************************************
* Si se define que se usen interrupciones, se crea un buffer para
* almacenar los datos recibidos, por default es de 64, para no ocupar
* mucho espacio en memoria RAM. *
**************************************************************************/
#if defined(USE_USART_INTERRUPTS)
#define USE_USART_RX_INTERRUPT
#define USE_USART_TX_INTERRUPT
#endif

#if defined(USE_USART_RX_INTERRUPT)
#include <avr/interrupt.h>
#define MAX_IN_BUFFER 64
volatile char _rx_buffer[MAX_IN_BUFFER];
volatile unsigned int _headrx;
volatile unsigned int _tailrx;
#endif
```

```
#if defined(USE_USART_TX_INTERRUPT)
#include <avr/interrupt.h>
#define MAX_OUT_BUFFER 64
volatile char _tx_buffer[MAX_OUT_BUFFER];
volatile unsigned int _headtx;
volatile unsigned int _tailtx;
volatile unsigned char _txbusy;
#endif

/************************************************************************
* Si se define el uso de la entrada salida estandar se declaran las
* funciones base y el stream que necesita la librería stdio.h para
* enviar y leer un dato mediante las funciones de la USART.
************************************************************************/
#if defined(USE_USART_STDIO)
#include <stdio.h>
static int _usart_putchar(char c, FILE *stream);
static int _usart_getchar(FILE *stream);
static FILE _myiostream = FDEV_SETUP_STREAM(_usart_putchar, _usart_getchar,
_FDEV_SETUP_RW);
#endif

/************************************************************************
* Implementacion de las funciones de la librería
************************************************************************/

/************************************************************************
* USART_Init
*
* Programa la USART del AVR con una configuración de 8 bits de palabra,
* sin paridad y un bit de stop a una velocidad "baudrate".
* Tambien habilita las interrupciones de recepcion de dato si se defie
```

```
* USE_USART_INTERRUPT y el uso de entrada y salida estandar si se define
* USE_USART_STDIO.
*
* Recibe:
* -baudrate: Velocidad en baudios a la que trabajará la USART.
*
*
**************************************************************************/
void USART_Init(long baudrate) {

USART_Init2(baudrate, USART_STOP_BITS_1,
USART_PARITY_OFF,
USART_DATA_SIZE_8);

}

/**************************************************************************
* USART_Init2
*
* Programa la USART del AVR con la confguracion especificada por el
* programador. A diferencia de USART_Init, se debe especificar el
* tamaño de palabra, la paridad, los bits de stop y el baudaje.
* Tambien habilita las interrupciones de recepcion de dato si se hace
* alguna definición de ellas y el uso de entrada y salida estandar si se define
* USE_USART_STDIO.
*
* Recibe:
*
* baudrate: Velocidad en baudios a la que trabajará la USART.
* strop_bits: Numero de bits de parada
* USART_STOP_BITS_1 - 1 Bit de parada.
* USART_STOP_BITS_2 - 2 Bits de parada.
```

```
* parity: Paridad.
* USART_PARITY_OFF - Sin paridad
* USART_PARITY_EVEN_ON - Paridad Par
* USART_PARITY_ODD_ON - Paridad Impar
* data_size: Tamaño de palabra
* USART_DATA_SIZE_5 - Tamaño de palabra de 5 bits
* USART_DATA_SIZE_6 - Tamaño de palabra de 6 bits
* USART_DATA_SIZE_7 - Tamaño de palabra de 7 bits
* USART_DATA_SIZE_8 - Tamaño de palabra de 8 bits
*
**********************************************************************/
void USART_Init2(long baudrate,
unsigned char stop_bits,
unsigned char parity,
unsigned char data_size){

/*Se programa el valor del baudrate segun la formula del datasheet*/
UBRRL = MYUBRR(baudrate);
UBRRH = MYUBRR(baudrate) >> 8;

UCSRC = _BV(URSEL) | stop_bits | parity | data_size;

/*Habilitamos el transmisor y el receptor*/
UCSRB = _BV(RXEN) | _BV(TXEN);

#if defined(USE_USART_RX_INTERRUPT)
UCSRB |= _BV(RXCIE);
_headrx=0;
_tailrx=0;
#endif

#if defined(USE_USART_TX_INTERRUPT)
UCSRB |= _BV(TXCIE);
```

```
_headtx=0;
_tailtx=0;
_txbusy=0;
#endif

#ifdef USE_USART_STDIO
stdout = stdin = & _myiostream;
#endif

}

/*****************************************************************************
* USART_PutChar
*
* Envia un byte por el puerto serie.
*
* Recibe:
*
* - byte: Dato a enviar por el puerto serie.
*****************************************************************************/
void USART_PutChar(char c) {

#if defined(USE_USART_TX_INTERRUPT)

if(!_txbusy) {

_txbusy=1;
UDR = c;

}
else {

UCSRB &= ~_BV(TXCIE);

_tx_buffer[_tailtx++] = c;

if(_tailtx>MAX_OUT_BUFFER)
```

```
_tailtx=0;
UCSRB |= _BV(TXCIE);
}
#else
while (!(UCSRA & _BV(UDRE)));
UDR = c;
#endif
}
/*****************************************************************************
* USART_GetChar
*
* Lee el dato recibido por el puerto serie.
* Si se define USE_USART_INTERRUP el dato se leera
* del buffer de entrada, de lo contrario esperara a
* que un dato este disponible en el registro de entrada.
*
* Devuelve:
* El dato leido del puerto serie o del buffer de entrada.
*****************************************************************************/
int USART_GetChar() {
#if defined(USE_USART_RX_INTERRUPT)
char ret=-1;
if(_headrx != _tailrx) {
ret=_rx_buffer[_headrx++];
if(_headrx>MAX_IN_BUFFER)
_headrx=0;
}
```

```
return ret;
#else
while (!(UCSRA & _BV(RXC)));
return UDR;
#endif
}
/*****************************************************************************
* USART_PutString_rx_bu
*
* Envia una cadena de caracteres por el puerto serie.
*
* Recibe:
* - str: Apuntador a la cadena de caracteres.
*****************************************************************************/
void USART_PutString(char *str) {

while (*str)
USART_PutChar(*str++);
}

/*****************************************************************************
* USART_Kbhit
*
* Indica si hay algun dato disponible para leer del registro de
* recepcion. Si se define USE_USART_RX_INTERRUP, la funcion devolvera
* el numero de datos disponibles en el buffer de entrada, de lo
* contrario devolvera 1 si hay un dato listo en el registro de
* recepcion o 0 si aun no hay dato listo.

*****************************************************************************/
int USART_Kbhit() {
```

```
#if defined(USE_USART_RX_INTERRUPT)
return _tailrx - _headrx;
#else
return (UCSRA & _BV(RXC));
#endif

}

/****************************************************************************
* ISR(USART_RCX_vect)
*
* Funcion que atiende la interrupcion de dato recibido por el
* puerto serie. Almacena el dato recibido en el buffer de entrada
* e incrementa el contador de datos recibidos.
*
* Solo se usa si se define USE_USART_RX_INTERRUPT ó
USE_USART_INTERRUPTS
****************************************************************************/
#if defined(USE_USART_RX_INTERRUPT)

ISR(USART_RXC_vect) {

_rx_buffer[_tailrx++]=UDR;

if(_tailrx>MAX_IN_BUFFER)
_tailrx=0;
}
#endif

/****************************************************************************
* ISR(USART_TCX_vect)
*
* Funcion que atiende la interrupcion transmisor vacío.
* Envia por el puerto serie el siguiente dato del buffer de salida.
```

```
*
* Solo se usa si se define USE_USART_TX_INTERRUPT ó
USE_USART_INTERRUPTS
*****************************************************************************/
#if defined(USE_USART_TX_INTERRUPT)

ISR(USART_TXC_vect) {

_txbusy=0;
if(_headtx != _tailtx) {

_txbusy=1;
UDR = _tx_buffer[_headtx++];
if(_headtx>MAX_OUT_BUFFER)
_headtx=0;

}

}
#endif

/*****************************************************************************
* Funciones para el uso de la entrada y la salida estandar con la
* USART.
*****************************************************************************/

#ifdef USE_USART_STDIO

/*****************************************************************************
* Funcion para el envio de un caracter mediante la salida estandar.
*****************************************************************************/
static int _usart_putchar(char c, FILE *stream) {
if(c=='\n')
_usart_putchar('\r',stream);
USART_PutChar(c);
```

```
return 0;
}
/*****************************************************************************
* Funcion para la recepcion de un caracter mediante la entrada estandar.
*****************************************************************************/
static int _usart_getchar(FILE *stream) {
char incomming;
while(!USART_Kbhit());

if((incomming=USART_GetChar())==13)
incomming='\n';
_usart_putchar(incomming,stream);

return incomming;
}

#endif

#endif /* USART_H_ */
```

7. Resultados

Al terminar el diseño, manufactura en el caso del sistema mecánico y el desarrollo de los diseños de las PCBs, se continúo con la elaboración de las placas por el método de planchado, que consta de imprimir en impresora láser los diseños sobre papel Couché y a continuación plancharlos sobre las placas fenólicas sin mover la hoja, cuidando de que el lado impreso de la hoja quede en contracara con el cobre, ya que esta se enfría se vierte en agua para poder remover el papel y solo queden las pistas pegadas a la placa, los resultados antes del baño acido se aprecian en las siguientes imágenes.

Imagen 7.1. Placas PCB impresas.

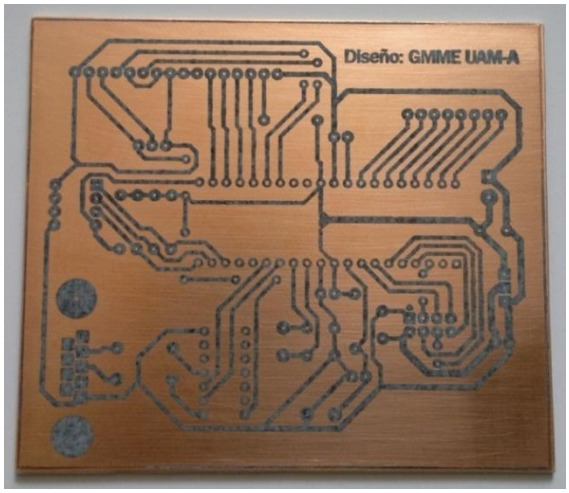


Imagen 7.2. Placa PCB del sistema maestro.

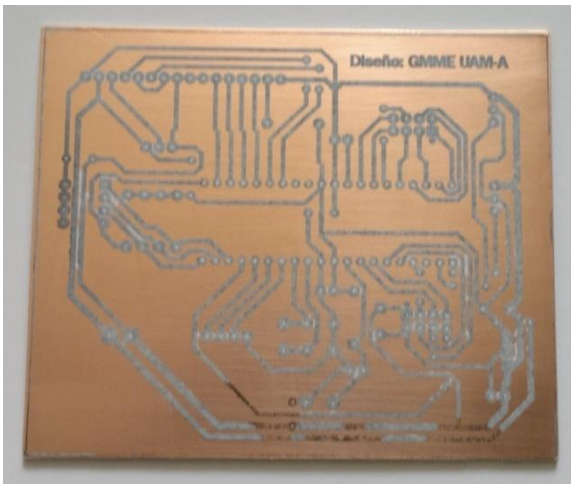


Imagen 7.3. Placa PCB del sistema esclavo.

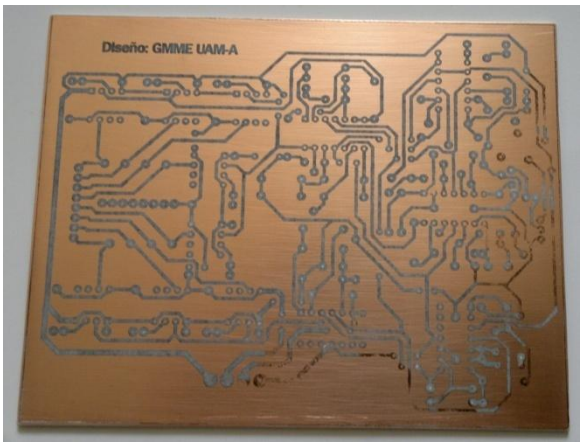


Imagen 7.4. Placa PCB del sistema de procesamiento de audio.

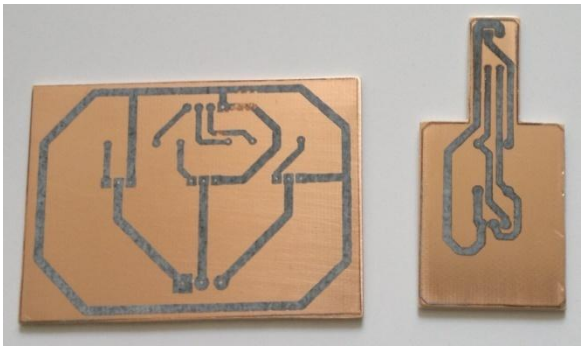


Imagen 7.5. Placas PCB del Driver de LEDs y del switch óptico del encoder.

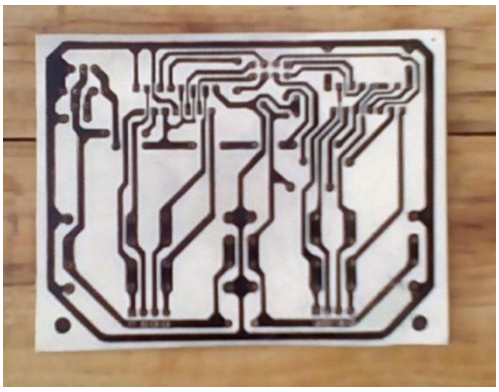


Imagen 7.6. Placa PCBdes pues del baño acido del Driver de los motores.

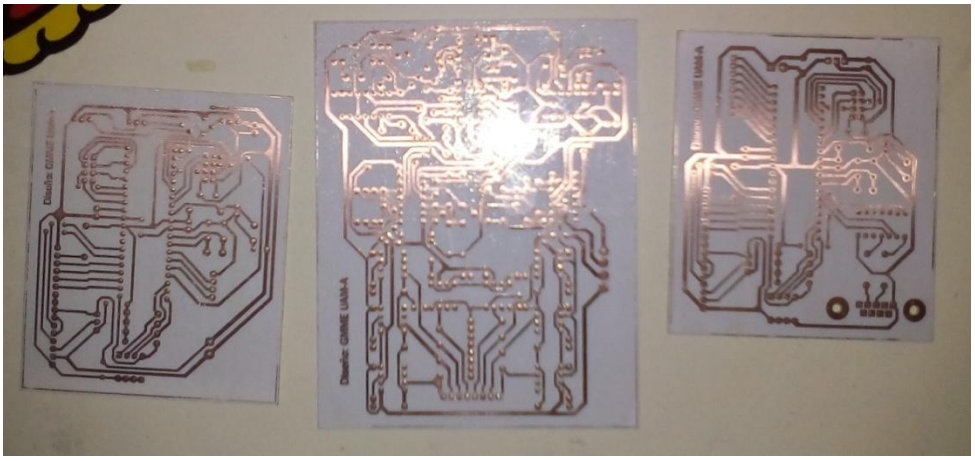


Imagen 7.7. Placas PCB después del baño acido.

Después del baño acido para quitar el cobre sobrante de las placas, se realizó el barrenado para instalar y soldar los componentes quedando se la siguiente manera.

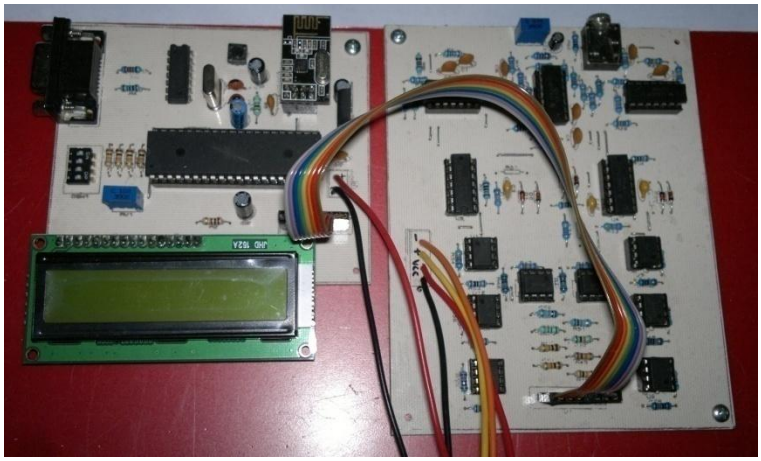


Imagen 7.8. Placas terminadas del sistema maestro.

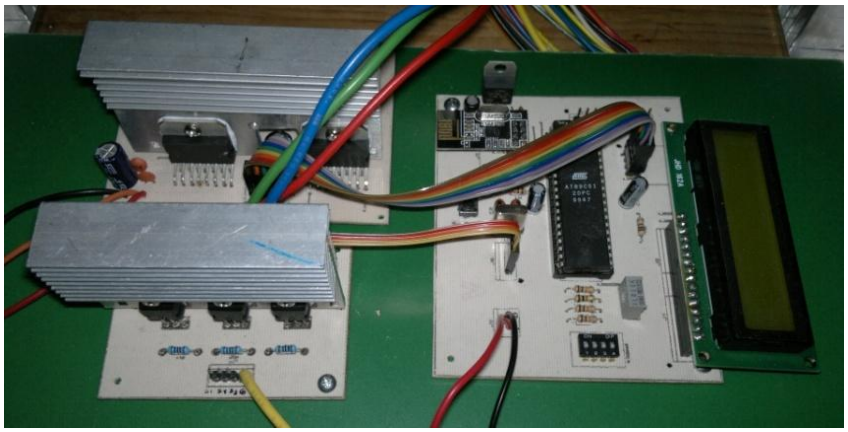


Imagen 7.9. Placas terminadas del sistema esclavo.

En el sistema mecánico, los resultados de la manufactura de las piezas que conforman el prototipo, son apreciables a la hora del ensamblado, ya que de existir errores o medidas incorrectas en las piezas no sería posible finalizar este proceso.

En las imágenes 7.10, 7.11 y 7.12 se aprecia el prototipo ensamblado desde distintas perspectivas.

Imagen 7.10. Prototipo ensamblado.

Imagen 7.11. Prototipo ensamblado.

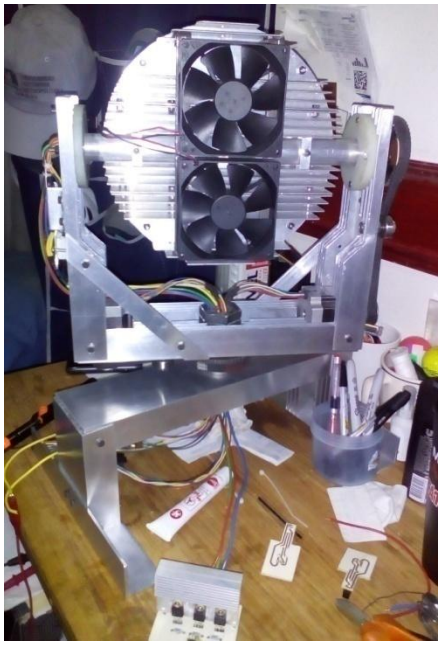


Imagen 7.12. Vista del prototipo.

En la imagen 7.13 se visualizan los ventiladores que ayudan a mantener en temperaturas óptimas a los LEDs de potencia.

Imagen 7.13. Vista trasera de la cabeza móvil.

Imagen 7.14. Vista lateral, sistema banda-poleas, eje vertical.

En la imagen anterior se aprecia montado el sistema de banda y poleas para transmitir los movimientos del motor a pasos al eje vertical y así hacer girar la cabeza del prototipo, también se visualiza un rodamiento fijo por un tornillo que cumple con la función de tensar la banda.

En las siguientes dos imagenes se aprecia en caso antes mencionado pero para el eje horizontal.

Imagen 7.15. Vista inferior, sistema banda-poleas, eje horizontal.

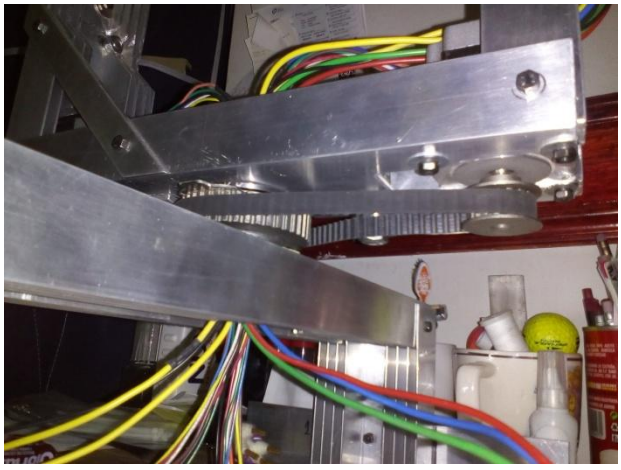


Imagen 7.16. Vista inferior desde otra perspectiva, eje horizontal.

Por último se podrá apreciar el encendido de los diodos LED de potencia, ubicados en la cabeza móvil del prototipo, distribuidos en tres grupos de colores, verde, rojo y azul, cada uno de 12 elementos, dando un total de 36 LEDs atribuyendo la característica de brindar iluminación en RGB.

Imagen 7.17. Vista del prototipo con los LEDs verdes encendidos.

Imagen 7.18. Vista del prototipo con los LEDs rojos encendidos.

Imagen 7.17. Vista del prototipo con los LEDs azules encendidos.

En general los resultados se sintetizan en la obtención de iluminación en distintos colores conocidos como RGB, con direccionamiento del haz de luz con el accionamiento de los motores a pasos derivado del procesamiento y las instrucciones generadas por las distintas etapas que conforman el prototipo.

8. Conclusiones

Durante el desarrollo de este proyecto se presentaron grandes retos por la falta de conocimientos en aéreas ajenas a la ingeniería electrónica, como lo fue todo lo relacionado con el sistema mecánico, principalmente en la manufactura al desconocer los procesos de maquinado y la utilización de la maquinaria, cabe mencionar que también se involucra el hecho de no saber donde adquirir el material, teniendo como consecuencia retrasos significativos y aumento en los costos por transportación, son aumentos mínimos que se ven reflejados de forma importante al termino del proyecto. También en todo lo relacionado con el desarrollo de la codificación y aplicación de algoritmos, al desconocer las plataformas y lenguajes de programación, principalmente cuando se trata de la implementación de nuevos módulos o sistemas por la escasa información existente, tanto en material bibliográfico como en internet.

Sin duda, lo antes mencionado nos incita a tomar ciertas consideraciones para el desarrollo de proyectos multidisciplinarios, en primera de contar con la asesoría de personal que cuente con los conocimientos necesarios en dichas aéreas, apoyo o integración con personas experimentadas en el desarrollo y/o uso de herramental ajeno a nuestras áreas de aplicación.

Con el maquinado de las piezas del sistema mecánico, se tuvo la dicha de adquirir nuevos conocimientos sobre la metodología a seguir para el uso de maquinaria afín a este proceso, el uso de ciertas herramientas e instrumental de manufactura.

Con el desarrollo del sistema electrónico, se reforzaron los conocimientos sobre diseño de sistemas electrónicos, el uso de la plataforma de simulación y se adquirió experiencia con la fabricación de las placas de circuito impreso, mejorando por mucho este proceso al igual que el procedimiento de soldadura de los componentes, derivando en un trabajo más pulcro y fino.

Por último, todo el trabajo realizado tanto mecánico como electrónico, no tendría mucha razón de ser sin la programación, ya que con este aspecto se le da vida a cualquier equipo, prototipo o sistema que cuente con un dispositivo de procesamiento de datos de uso general, como lo es un procesador o microcontrolador.

9. Referencias bibliográficas

[1] Dra. Carina Toxqui Quitl, “¿Qué es la Optomecatrónica?”, Profesora investigadora de la Universidad Politécnica de Tulancingo y Coordinadora del Doctorado en Optomecatrónica.
http://www.upt.edu.mx/Contenido/OfertaEducativa/Doctorado/InfOptomecatronica/Optomecatronica.pdf

[2] Ernesto Bechara, “Fichas Educativas de Iluminación A-prende La Luz! Nº 40. Sistemas Digitales 2: Luces Robóticas. Primera Parte”, Profesional del Espectáculo / Diseño de Iluminación / Docente.
http://www.becharaespectaculos.com.ar/index.html

[3] Carlos A. Reyes, “Microcontroladores PIC Programación en Basic”, 2a. Edición, Rispergraf

[4] Sergio Franco, “Diseño con amplificadores operacionales y circuitos integrados analógicos”, 3a. Edición, Mc Graw Hill

[5] MrElberni “USART AVR Comunicación serial”, http://microcontroladores-mrelberni.com

[6] Muhammad H. Rashid, Circuitos Microelectrónicos: Análisis y Diseño, I Edición, International Thomson Editors, 2000.

[7] Adel S. Sedra, Circuitos Microelectrónicos, IV Edición, Oxford University Press, 1998.

[8] Robert B. Northrop, Analog Electronic Circuits, I Edición, Addison-Wesley Publishing Company, 1990.

[9] Robert Spence, Tolerance Desing of Electronic Circuits, I Edición, Imperial College Press, 1988.

[10] Luis E. Avendaño, Sistemas Electrónicos Analógicos: Un Enfoque Matricial, Primera Edición, Centro de Publicaciones UTP, 2007.

[11] Cosas de ingeniería, “Transceiver nRF24L01 + módulo 2.4 GHz” http://cosasdeingenieria.com
ftp://imall.iteadstudio.com/Modules/IM120606002_nRF24L01_module/DS_nRF24L01.pdf

[12] Accademia della Luce “El lenguaje DMX512”, www.accademiadellaluce.it

[13] Control de motores paso a paso mediante microcontroladores (Stepper motor)
http://server-die.alc.upv.es/asignaturas/lsed/2002-03/MotoresPasoaPaso/Motorespasoapaso.pdf

[14] West Instruments de México, “Manual de aplicación de encoders”.
http://www.westmexico.com.mx

[15] “LEDs de POTENCIA”, http://www.siled.com.mx

[16] “Software Proteus”, https://www.labcenter.com

[17] Datasheet Atmega32, http://www.atmel.com/Images/2503S.pdf

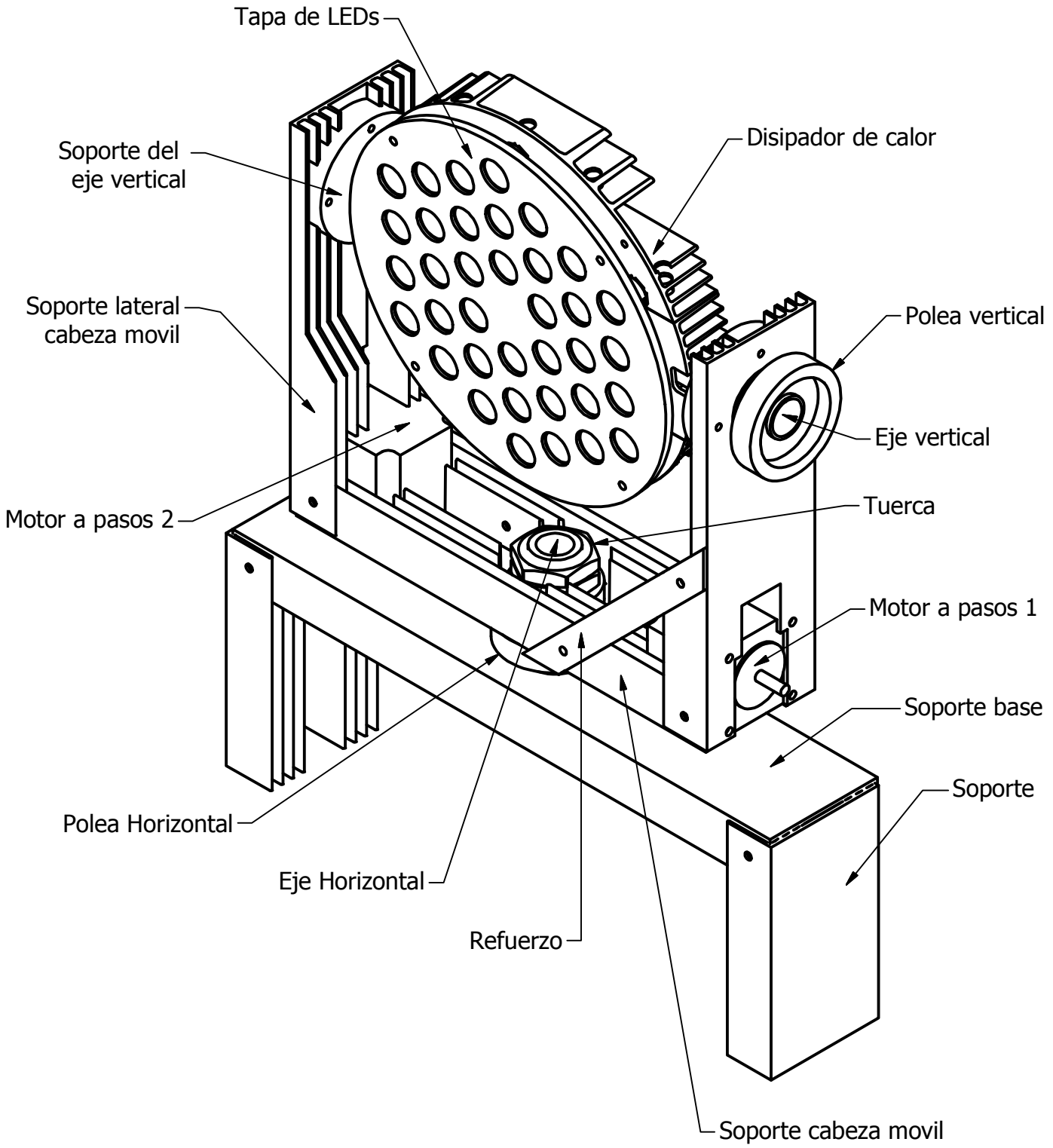
[18] Datasheet Atmega8535, http://www.atmel.com/Images/2502S.pdf

10. Entregables

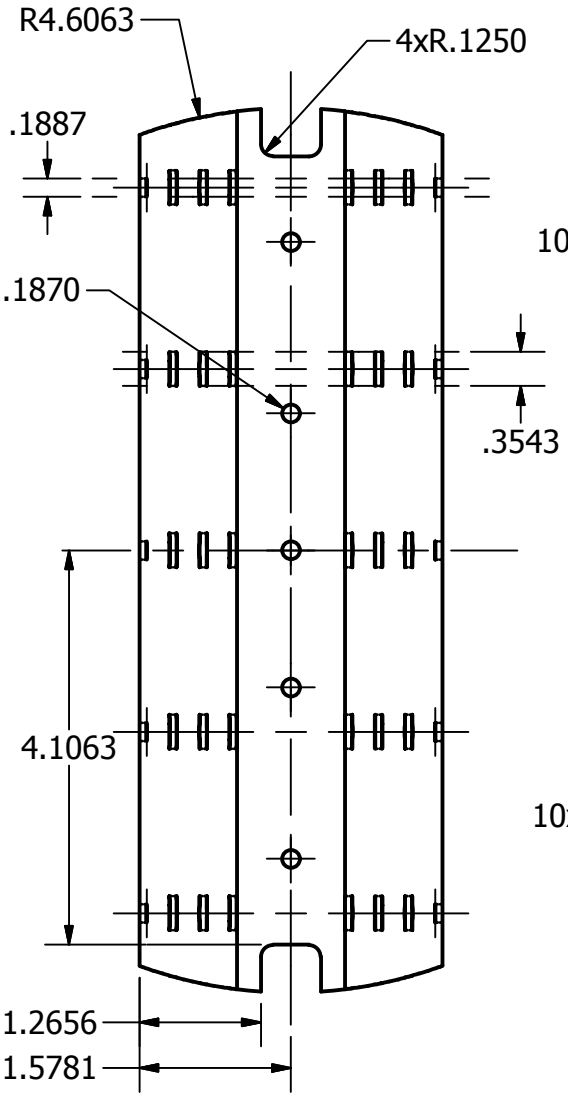
- Diagrama a bloques del prototipo
- Simulaciones con sus respectivos archivos.
- Diagramas eléctricos, de circuito impreso (PCB) y de mascara de componentes.
- Discos con software de interfaces y controladores del prototipo.
- Bitácora de trabajo.
- Demostración del prototipo funcionando.

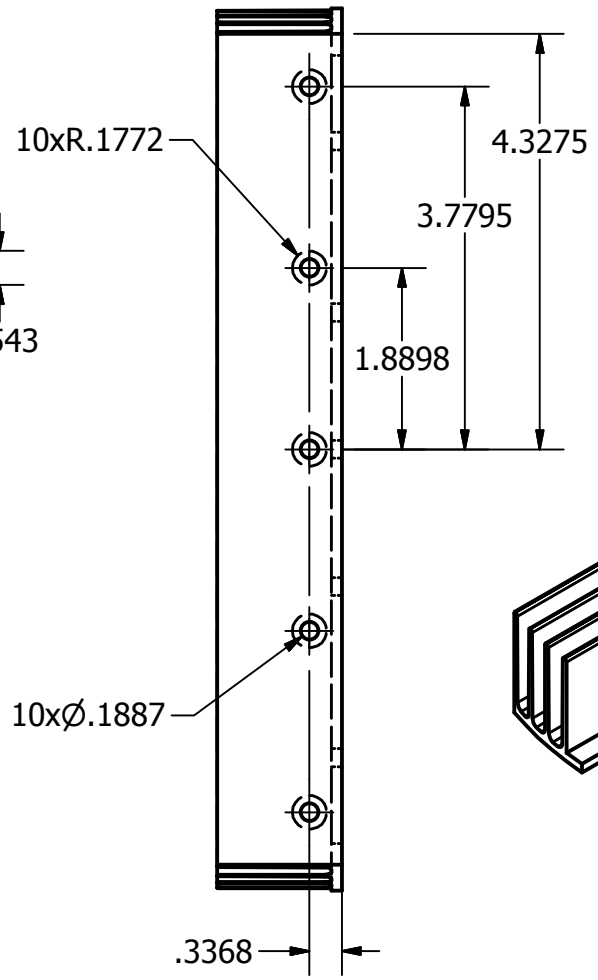
ANEXO A

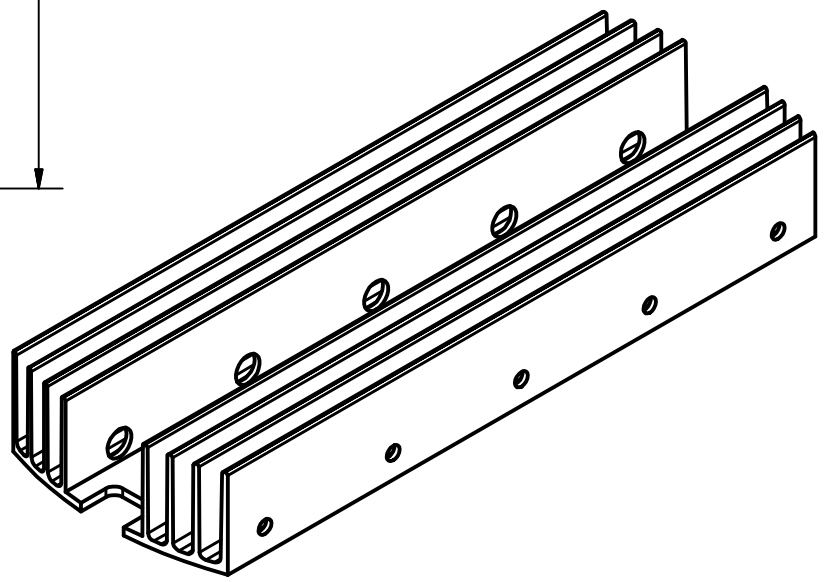
PLANOS 2D DE PIEZAS DEL SISTEMA MECÁNICO Y ESTRUCTURA DEL PROTOTIPO



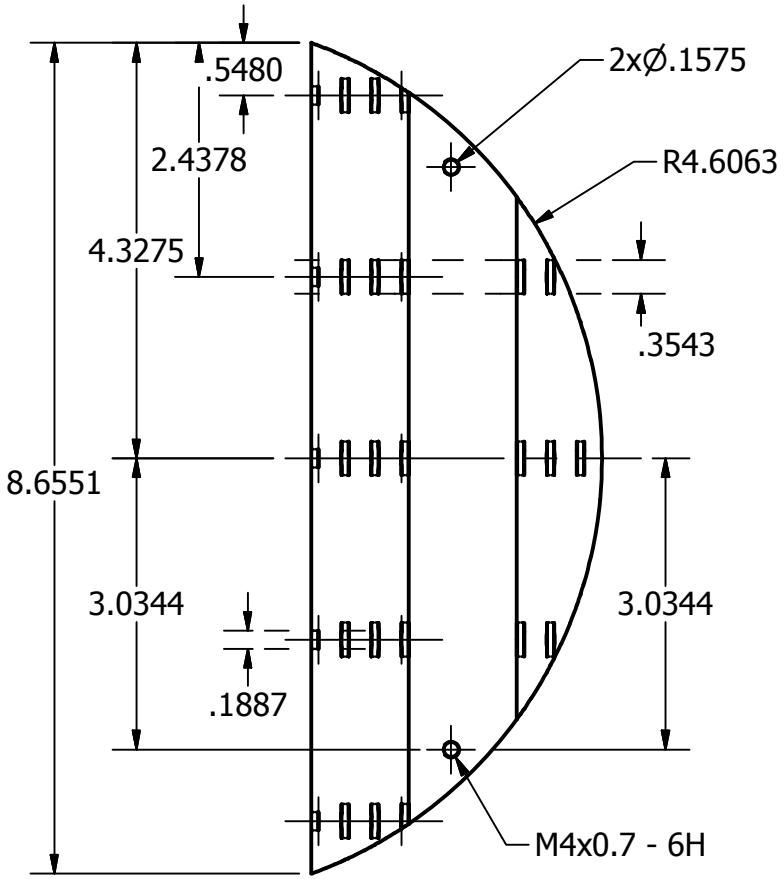
Escala: 1/3	UAM AZCAPOTZALCO	Dibujó:GADIEL M.
Acotacion: Pulg.		Revisó
	Prototipo	

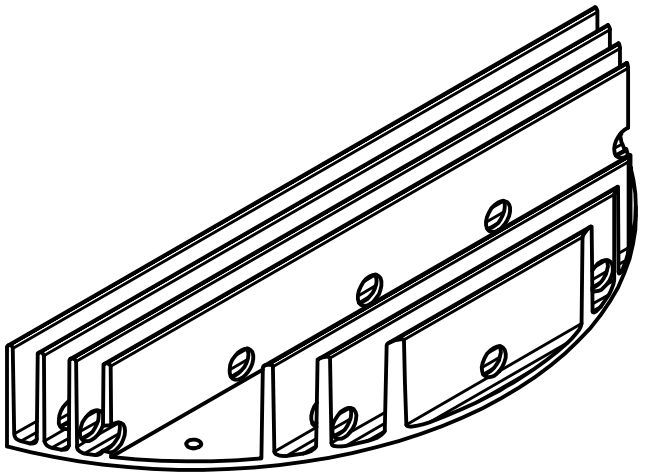


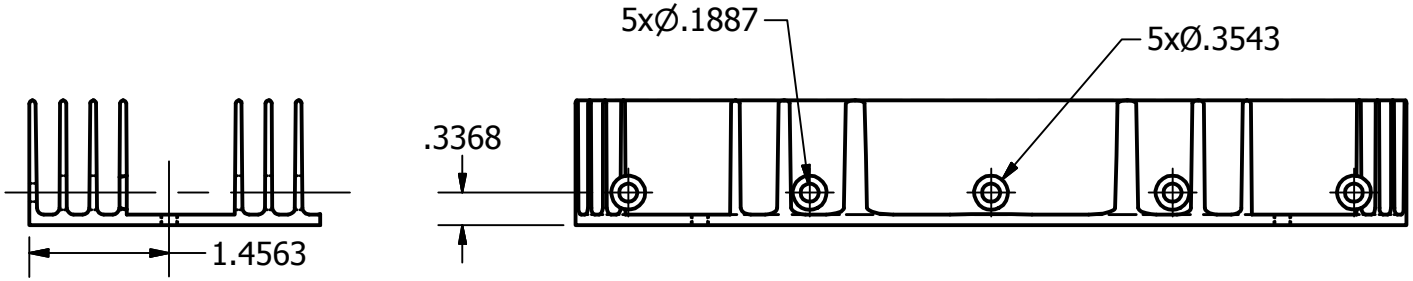




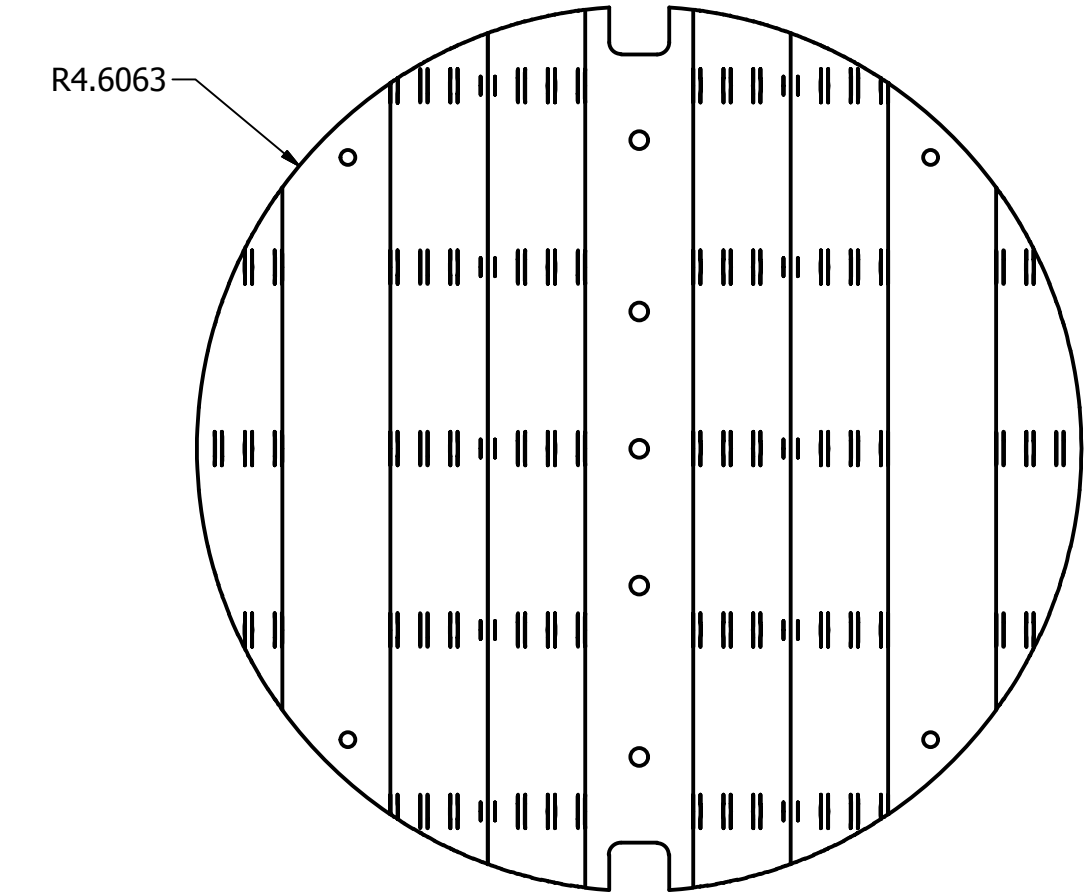
Escala: 1:2	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: pulg		Revisó
	DISIPADOR BARA CEN	

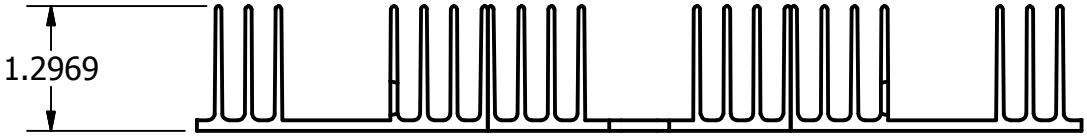




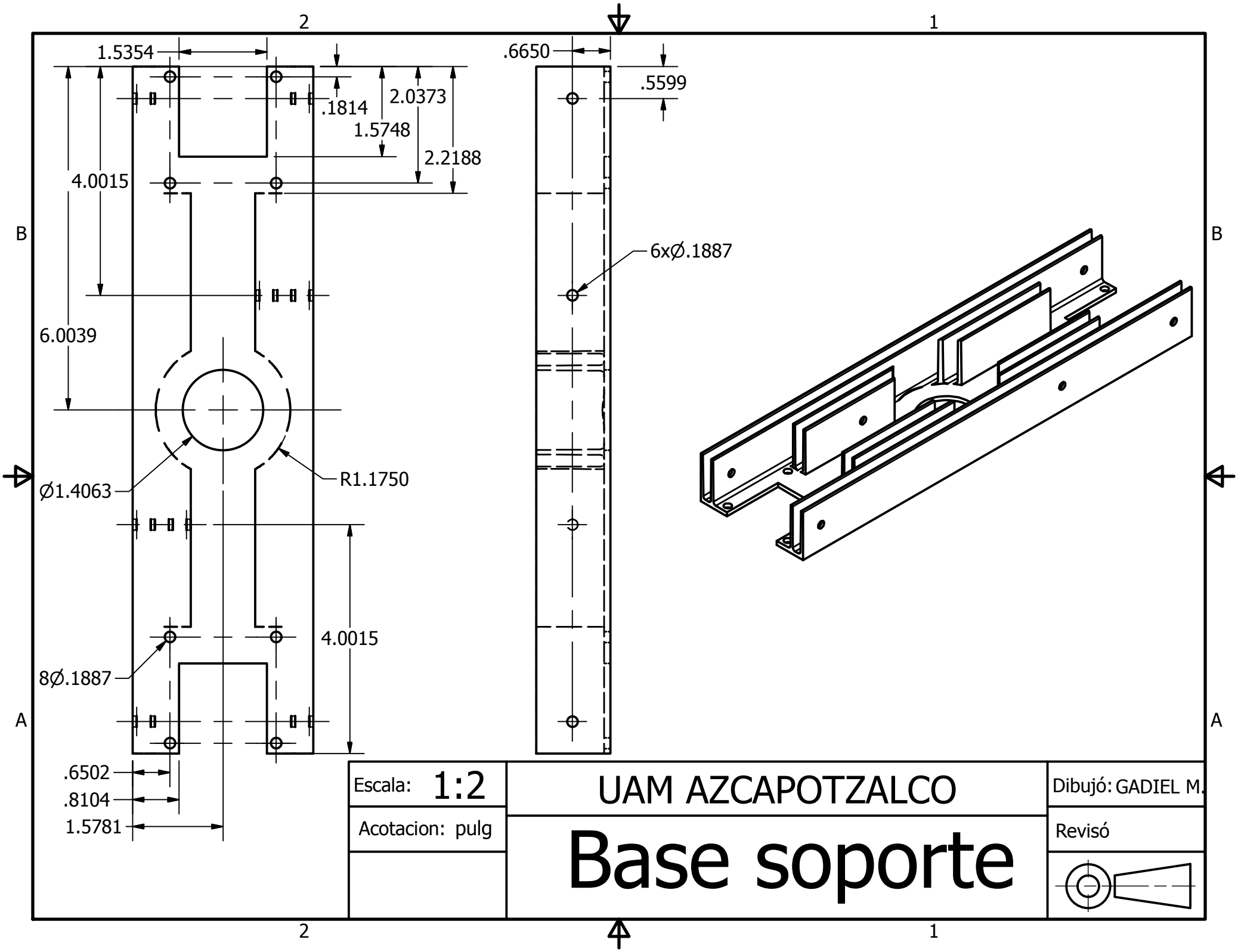


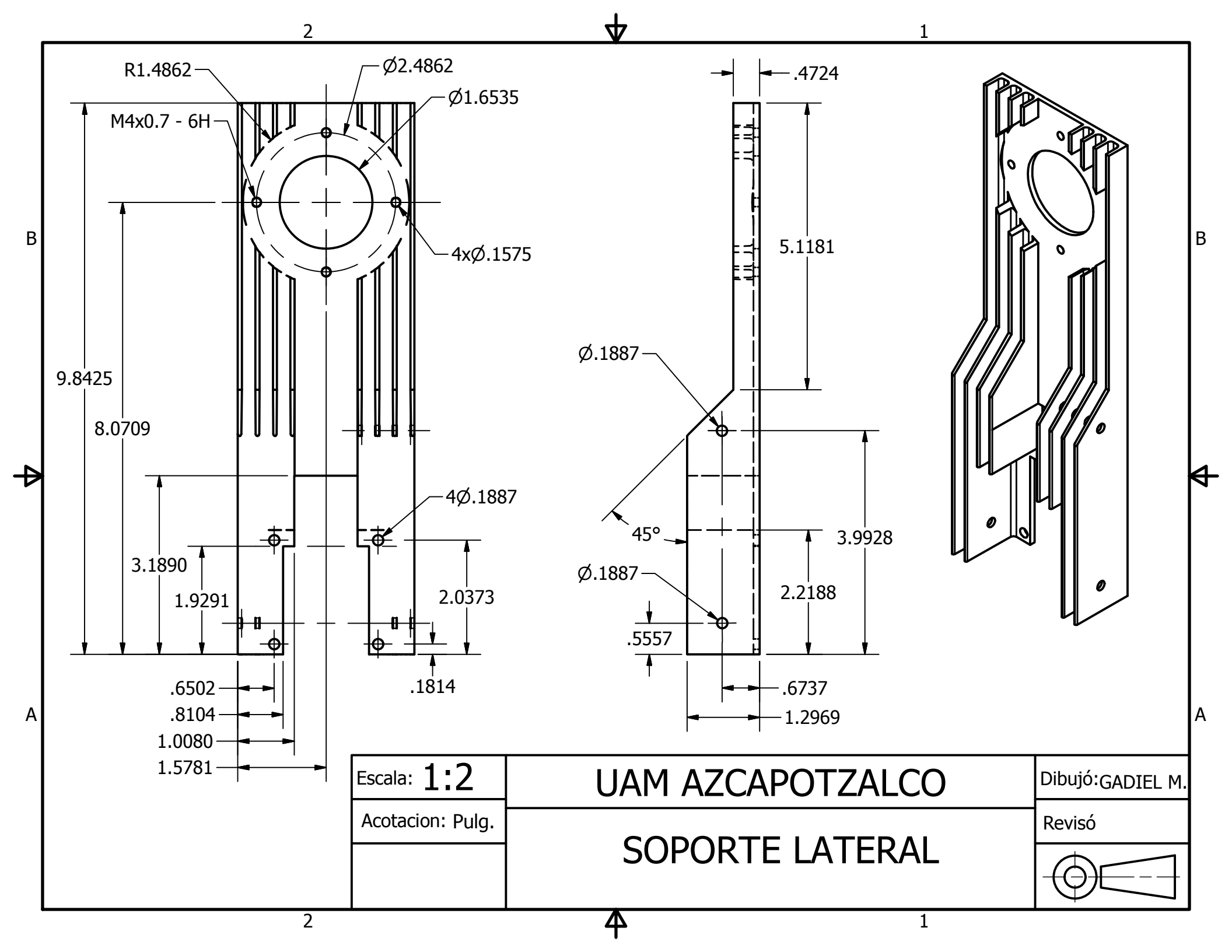
Escala: 1:2	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	DISIPADOR BARRA LAT 1 Y 2	Revisó

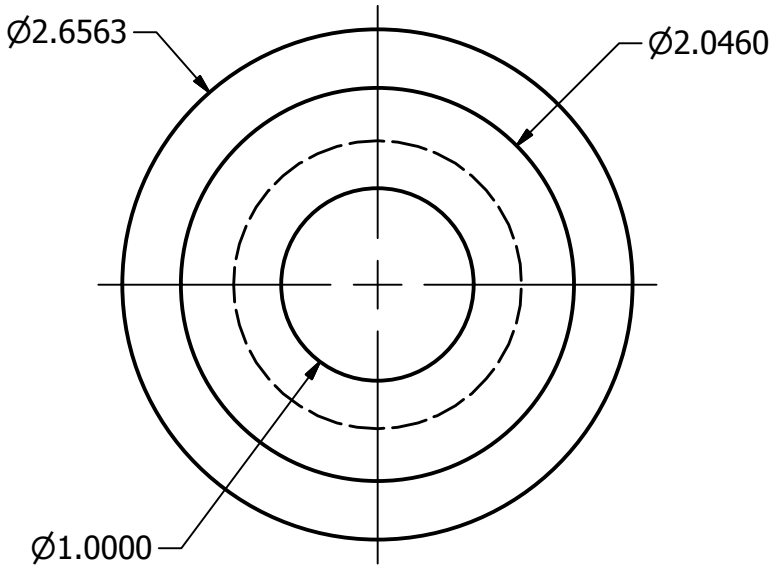


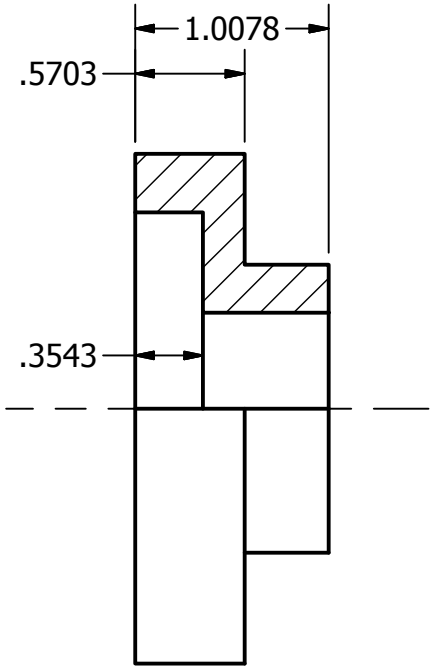


Escala: 1:2	UAM AZCAPOTZALCO	Dibujó:GADIEL M.
Acotacion: Pulg.		Revisó
	DISIPADOR	

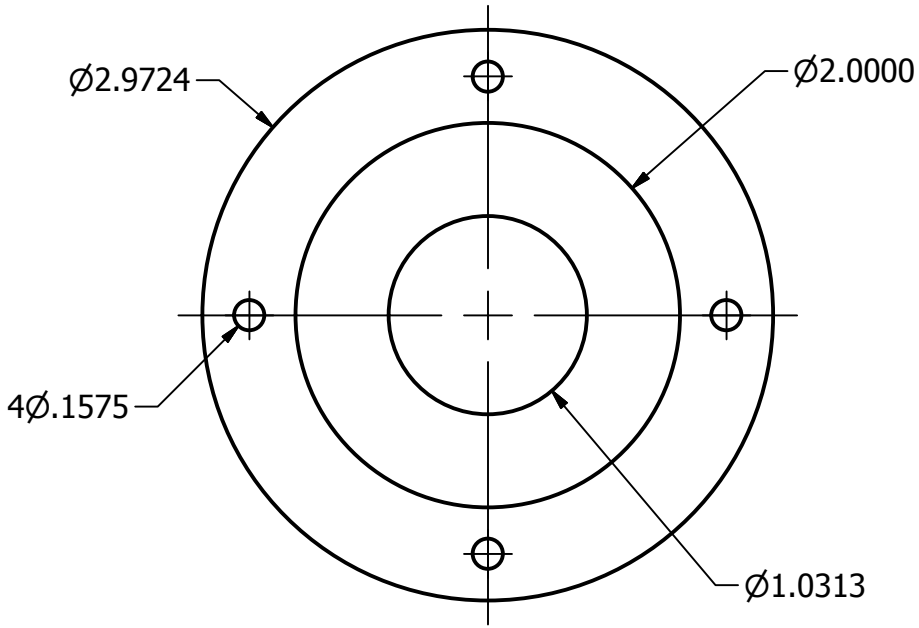

1.5354
.1814
2.0373
1.5748
2.2188
4.0015
6.0039
Ø1.4063
R1.1750
4.0015
8Ø.1887
.6502
.8104
1.5781
.6650
.5599
6xØ.1887
Escala: 1:2
Acotacion: pulg
UAM AZCAPOTZALCO
Base soporte
Dibujó: GADIEL M.
Revisó

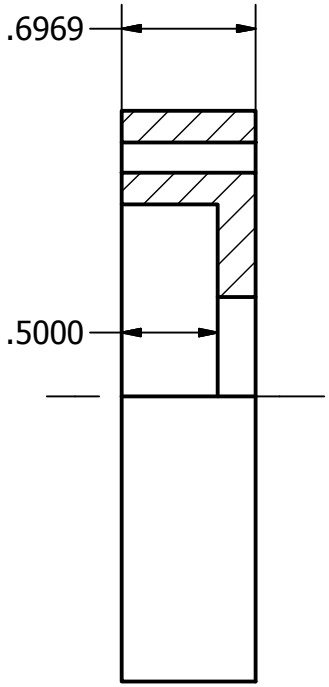

R1.4862
Ø2.4862
Ø1.6535
M4x0.7 - 6H
4xØ.1575
9.8425
8.0709
4Ø.1887
3.1890
1.9291
2.0373
.1814
.6502
.8104
1.0080
1.5781
.4724
5.1181
Ø.1887
45°
3.9928
Ø.1887
2.2188
.5557
.6737
1.2969
Escala: 1:2
Acotacion: Pulg.
UAM AZCAPOTZALCO
SOPORTE LATERAL
Dibujó: GADIEL M.
Revisó



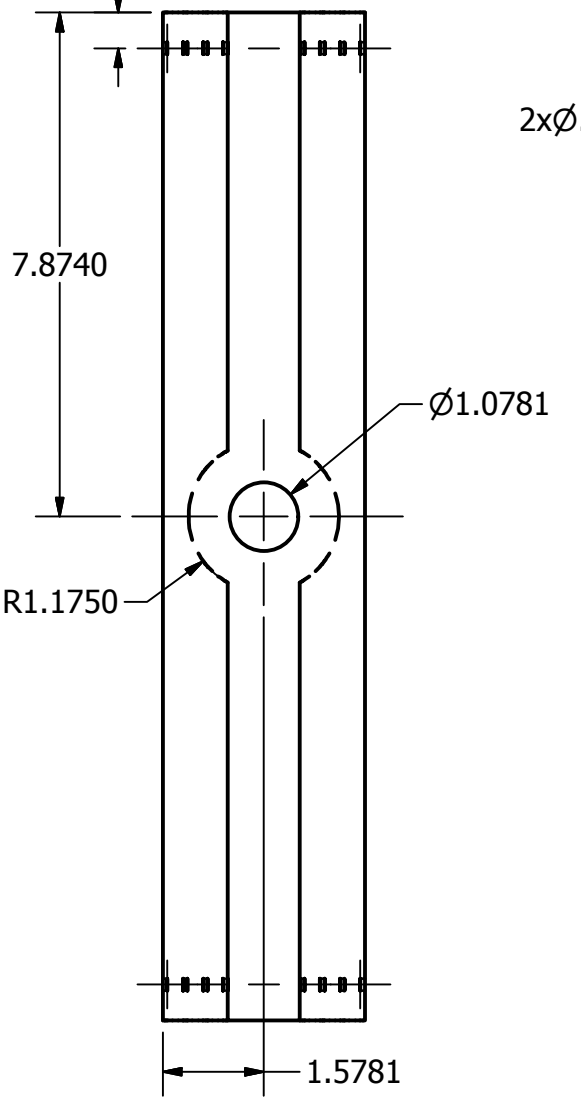


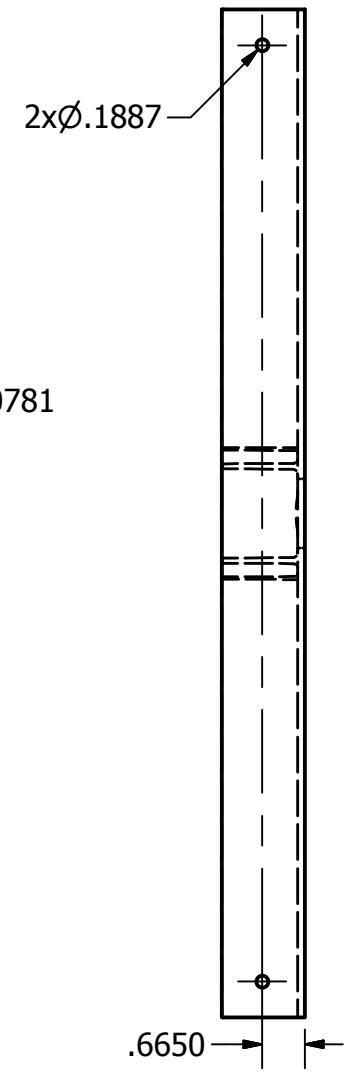
Escala: 1:1	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	POLEA L1	Revisó

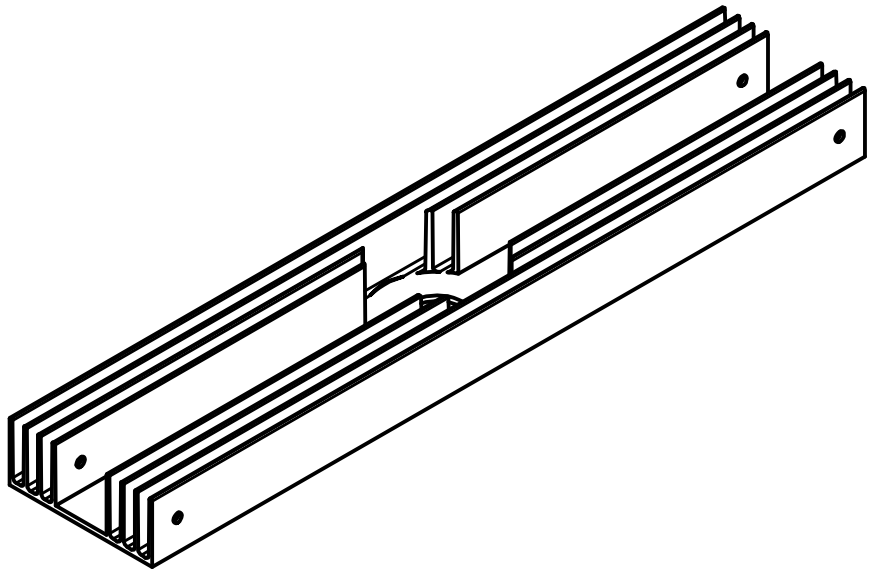




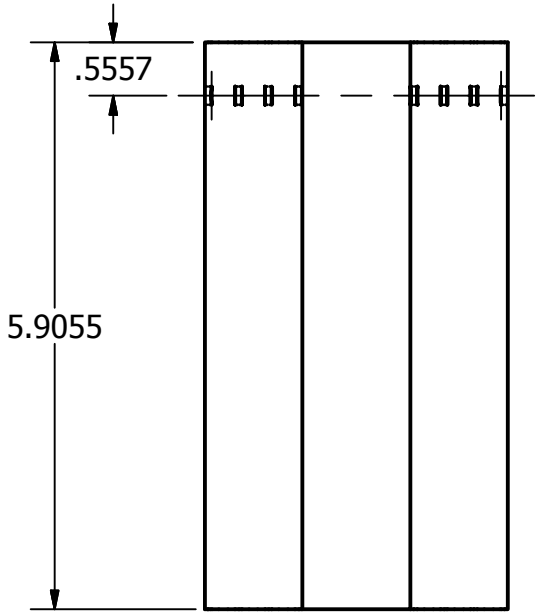
Escala: 1:1	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	ROTEJE	Revisó

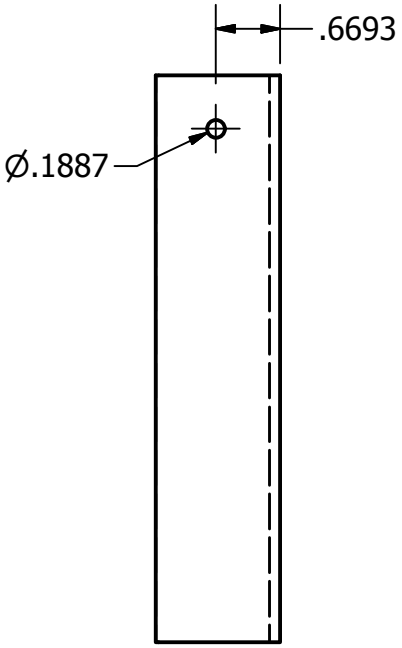


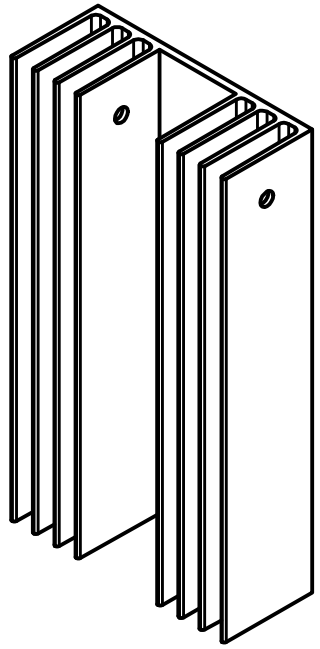




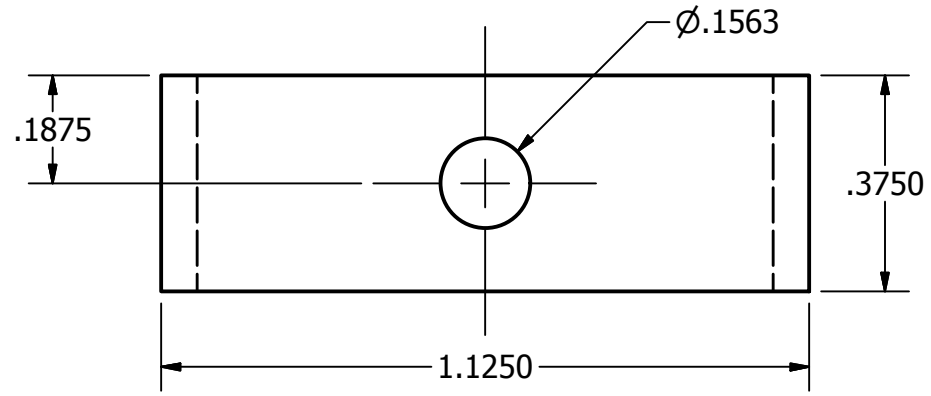
Escala: 1:3	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	SOPORTE 1	Revisó

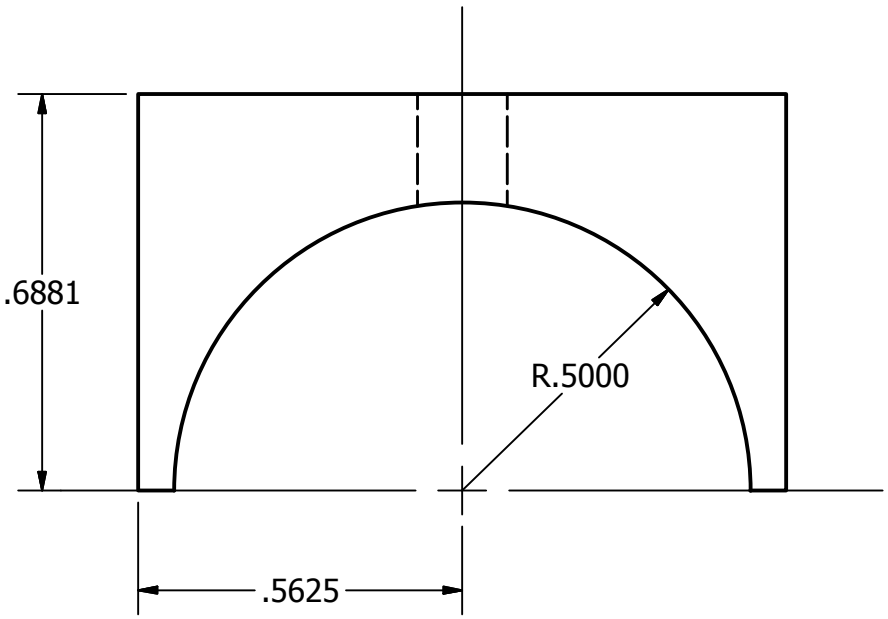




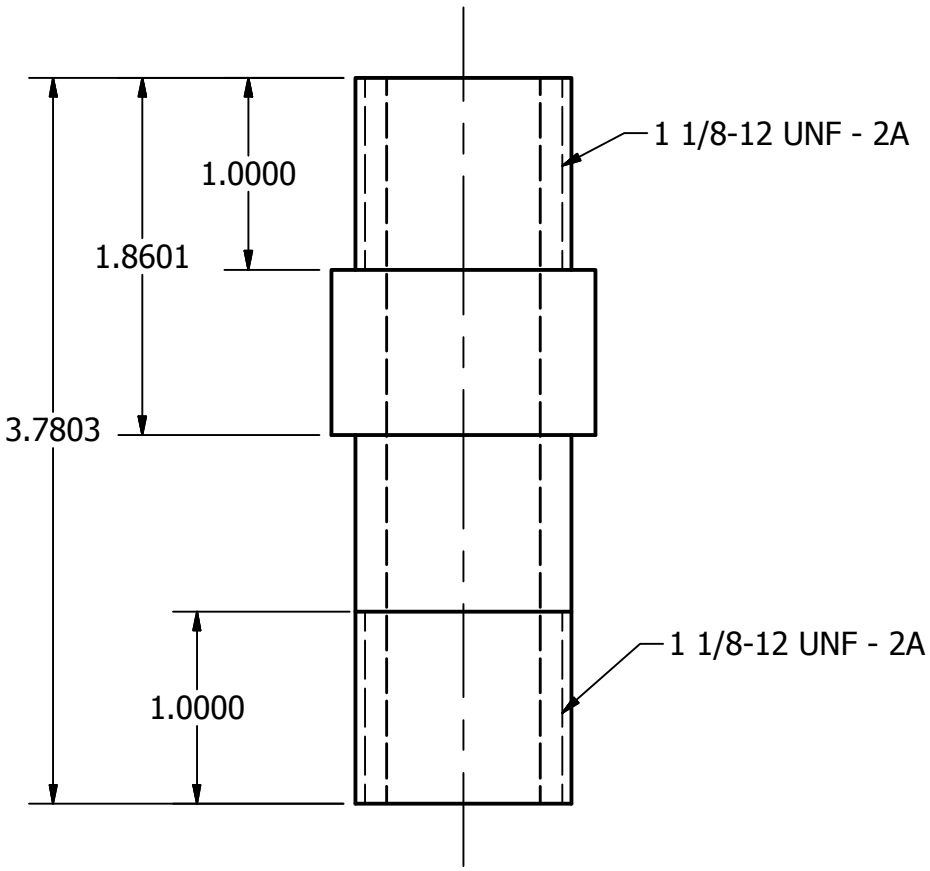


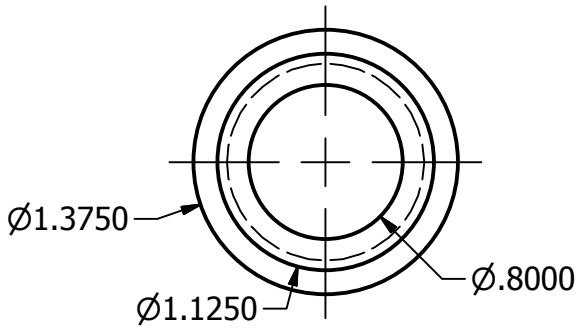
Escala: 1:2	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	SOPORTE 2	Revisó

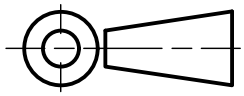


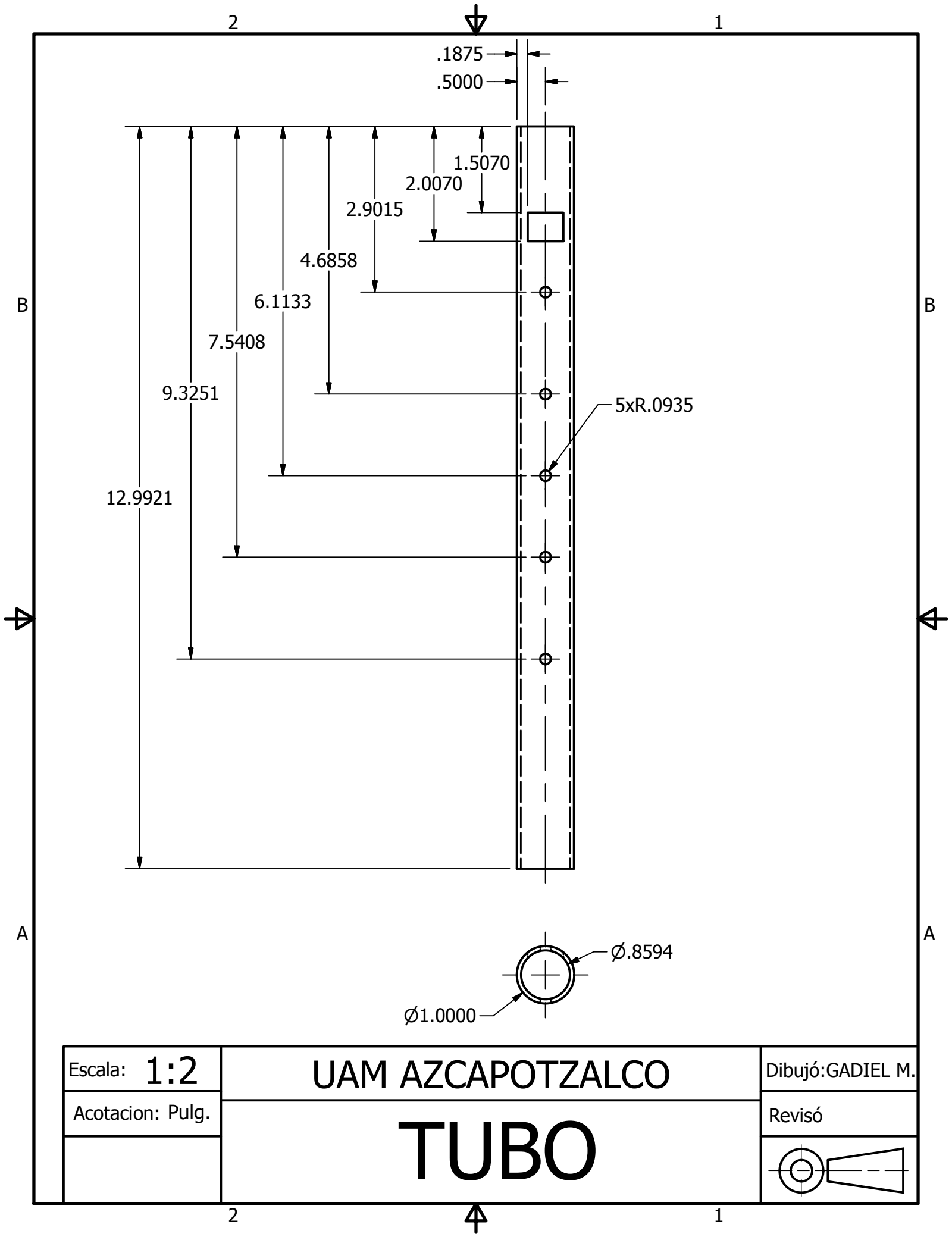


Escala: 3:1	UAM AZCAPOTZALCO	Dibujó:GADIEL M.
Acotacion: Pulg.	SUJETADOR	Revisó

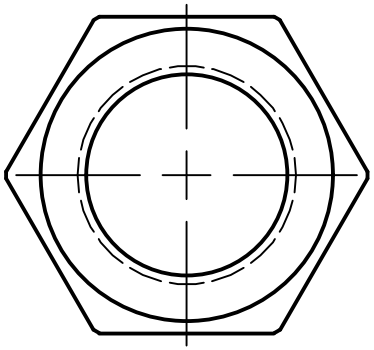


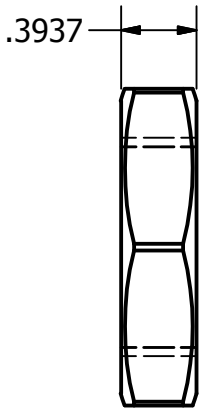


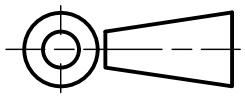
Escala: 1:1	UAM AZCAPOTZALCO	Dibujó:GADIEL M.
Acotacion: Pulg.	TUBO AXIAL	Revisó
		

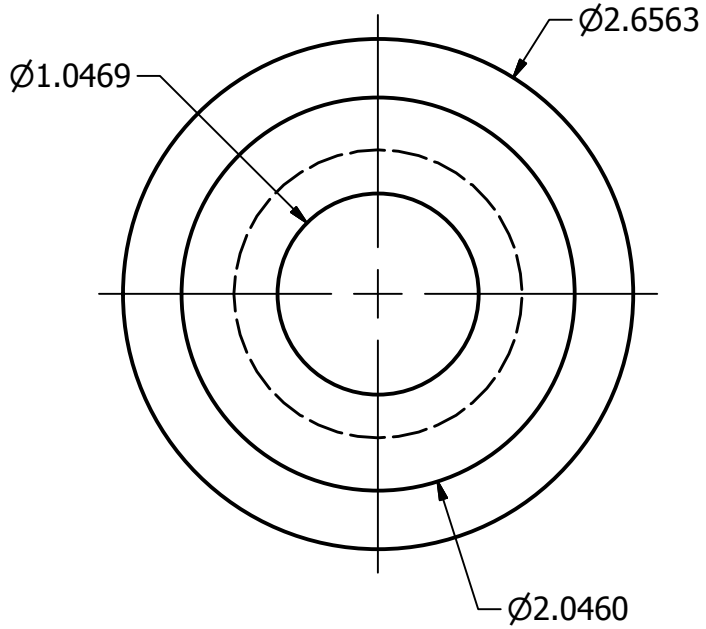


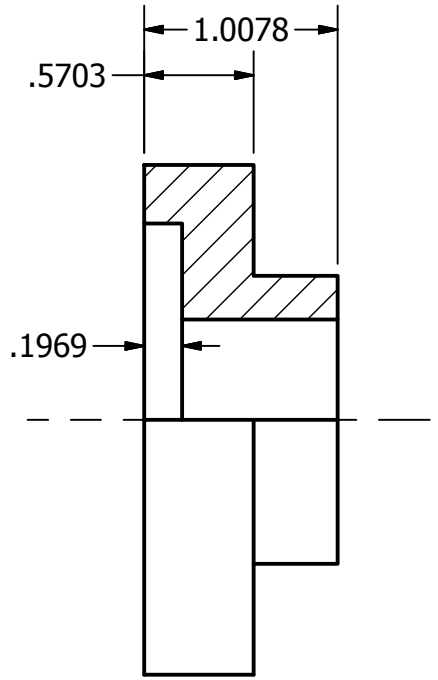
Escala: 1:2	UAM AZCAPOTZALCO	Dibujó:GADIEL M.
Acotacion: Pulg.	TUBO	Revisó



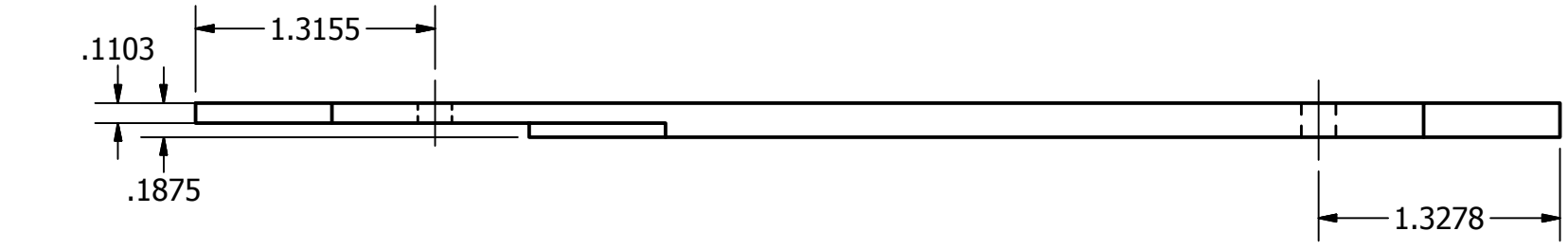


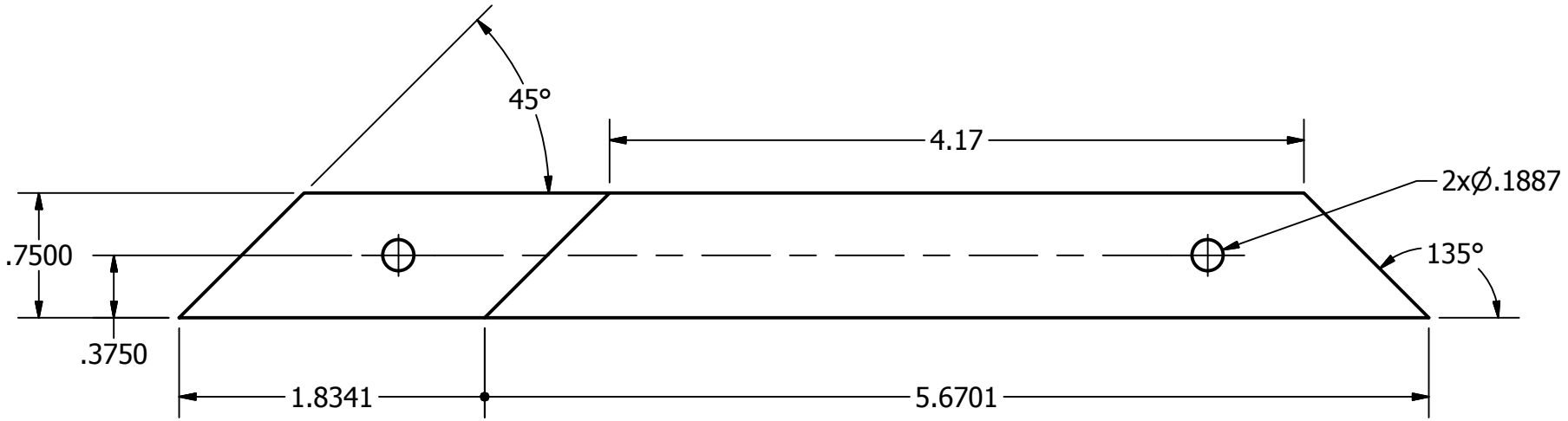
Escala: 1:3	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	CORTE DE TUERCA	Revisó
		



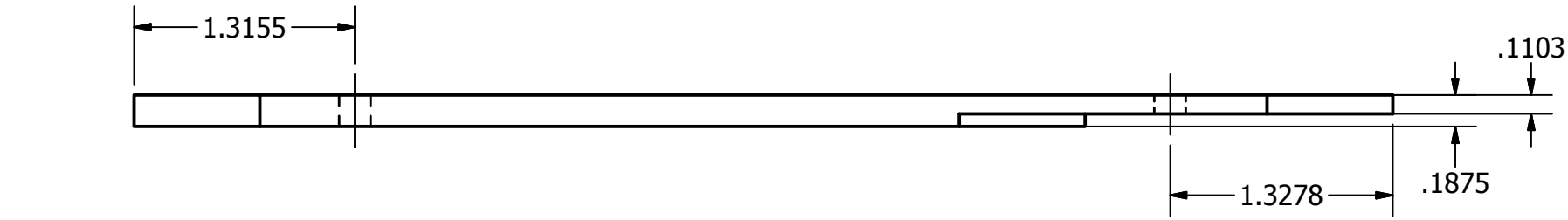


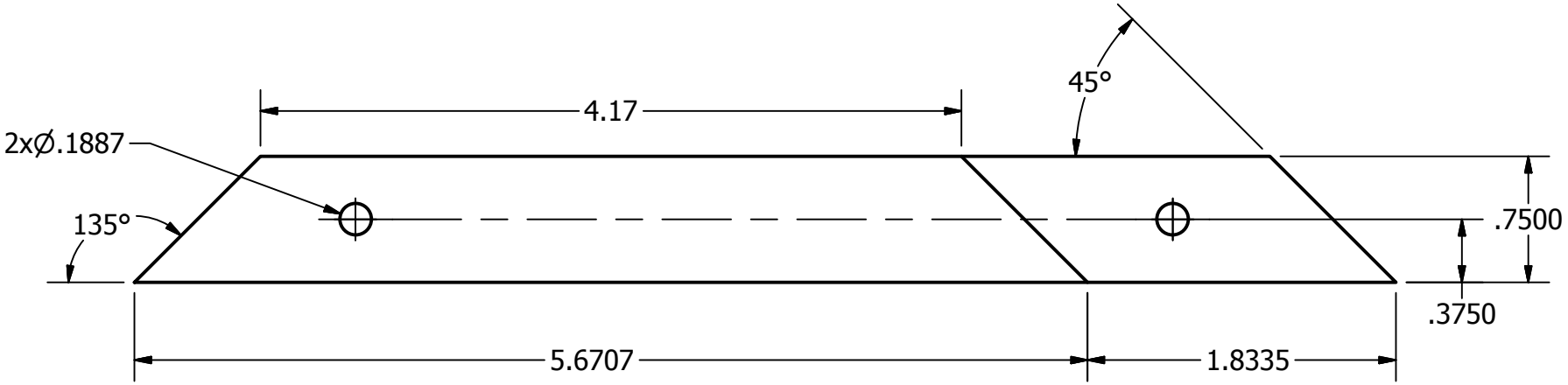
Escala: 1:1	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	POLEA B1	Revisó



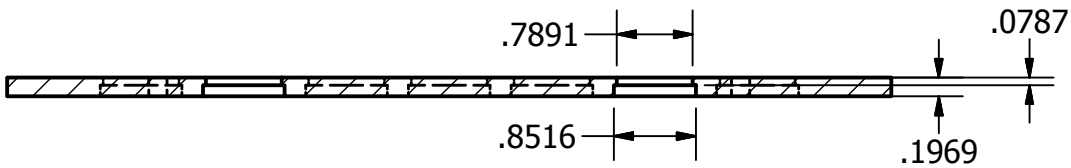


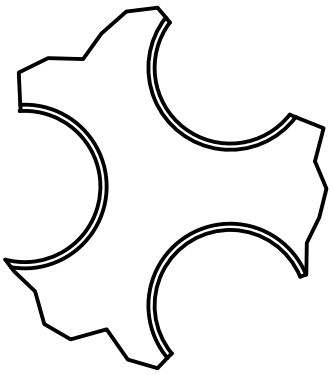
Escala: 1:1	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.	REFUERZO 2	Revisó



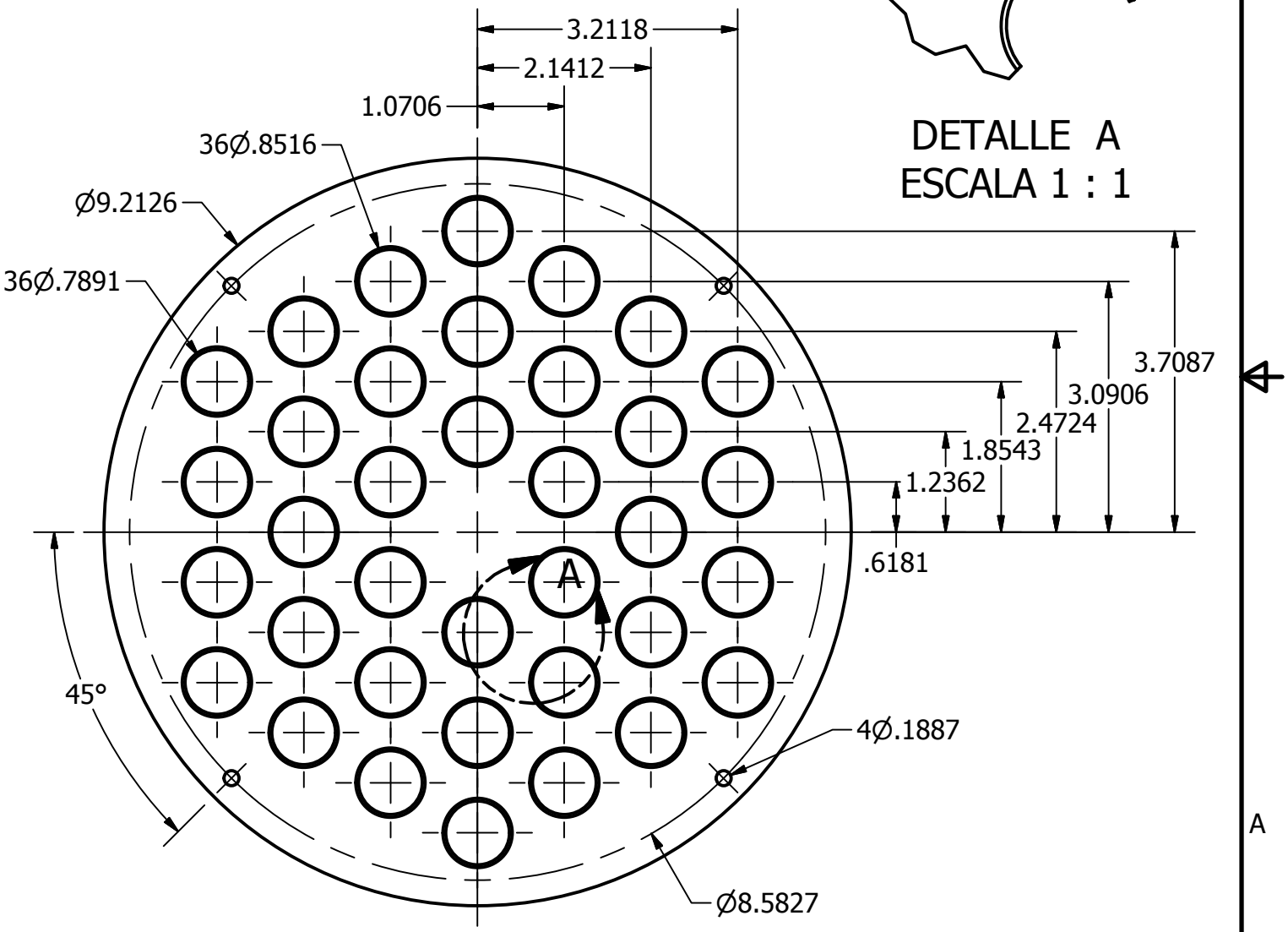


Escala: 1:1	UAM AZCAPOTZALCO	Dibujó: GADIEL M.
Acotacion: Pulg.		Revisó
	REFUERZO 1	





DETALLE A
ESCALA 1 : 1

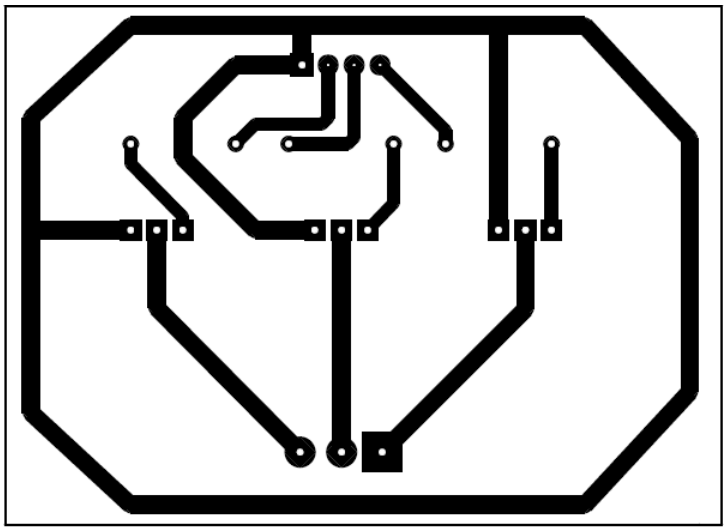


Escala: 1:2	UAM AZCAPOTZALCO	Dibujó:GADIEL M.
Acotacion: Pulg.	TAPA LEDS DE POTENCIA	Revisó

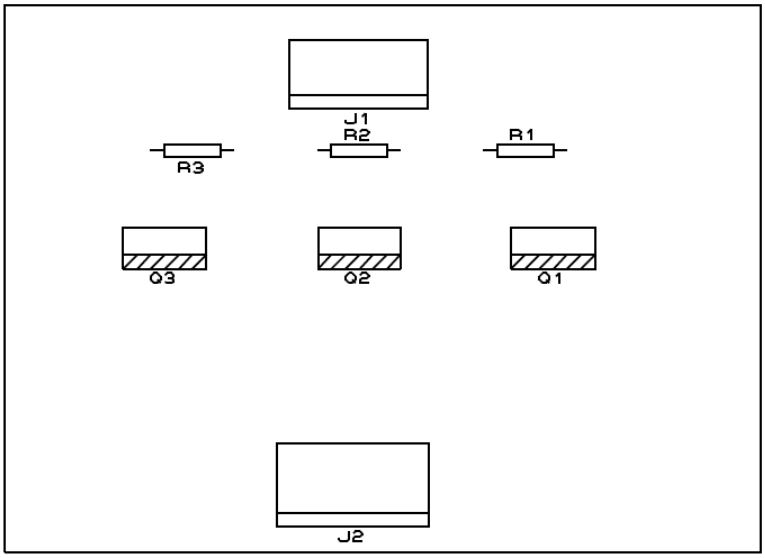
ANEXO B

DISEÑOS DE PLACAS DE CIRCUITO IMPRESO Y MASCARA DE COMPONENTES

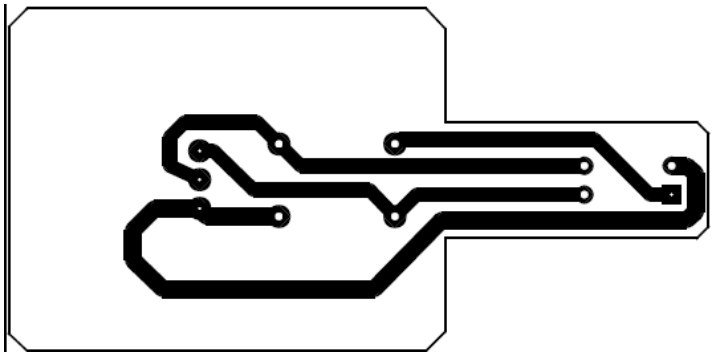
Circuito PCB del driver de los diodos LED



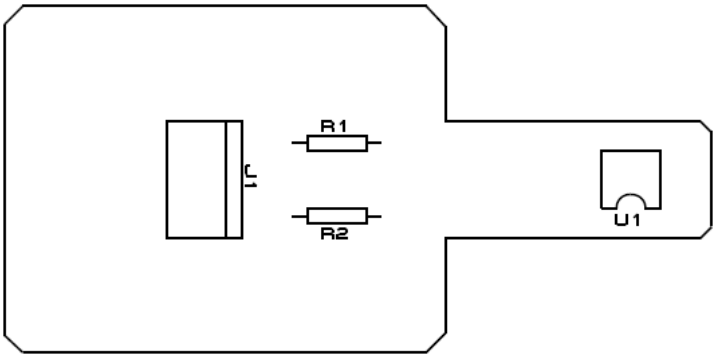
Mascara de componentes



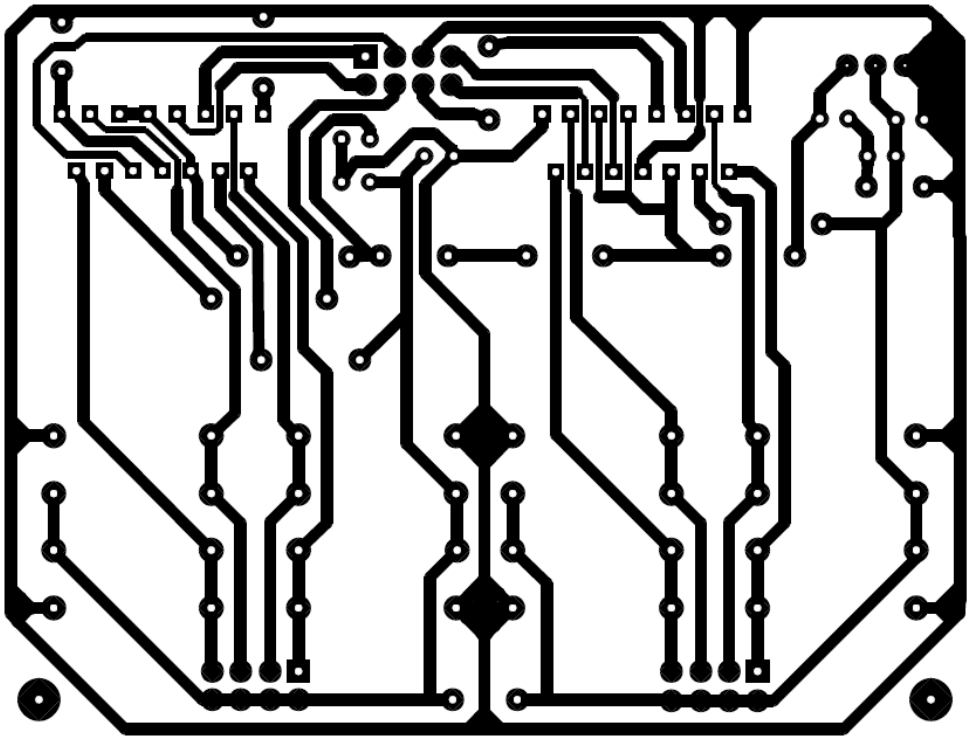
Circuito PCB del switch óptico para el encoder



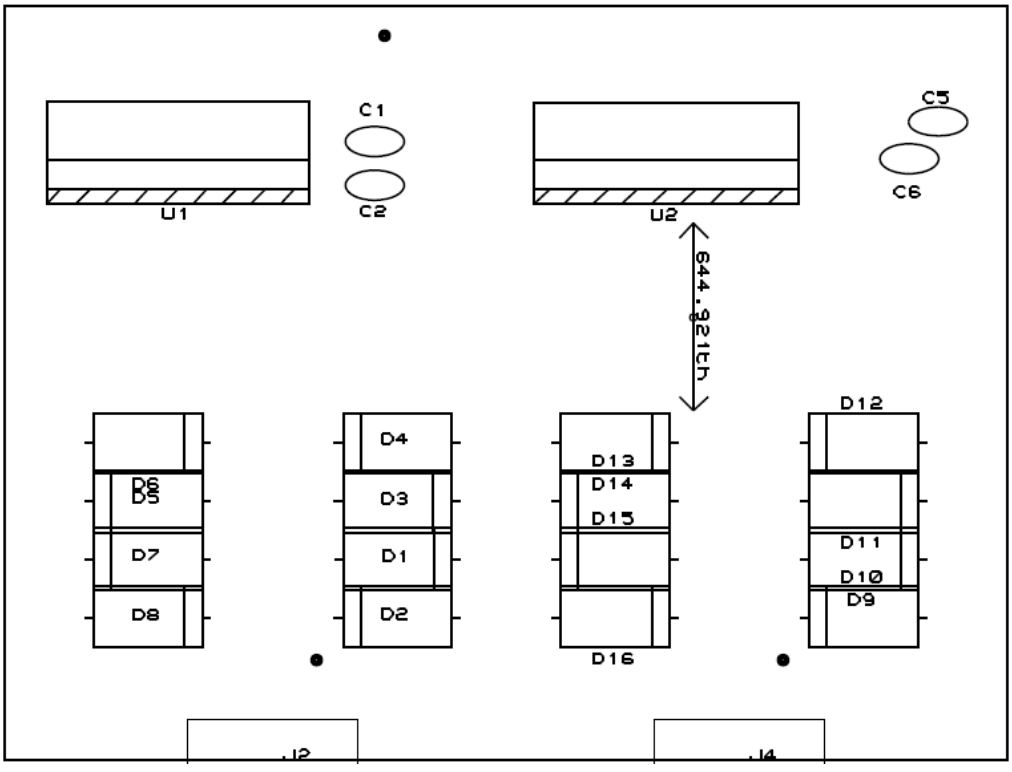
Mascara de componentes



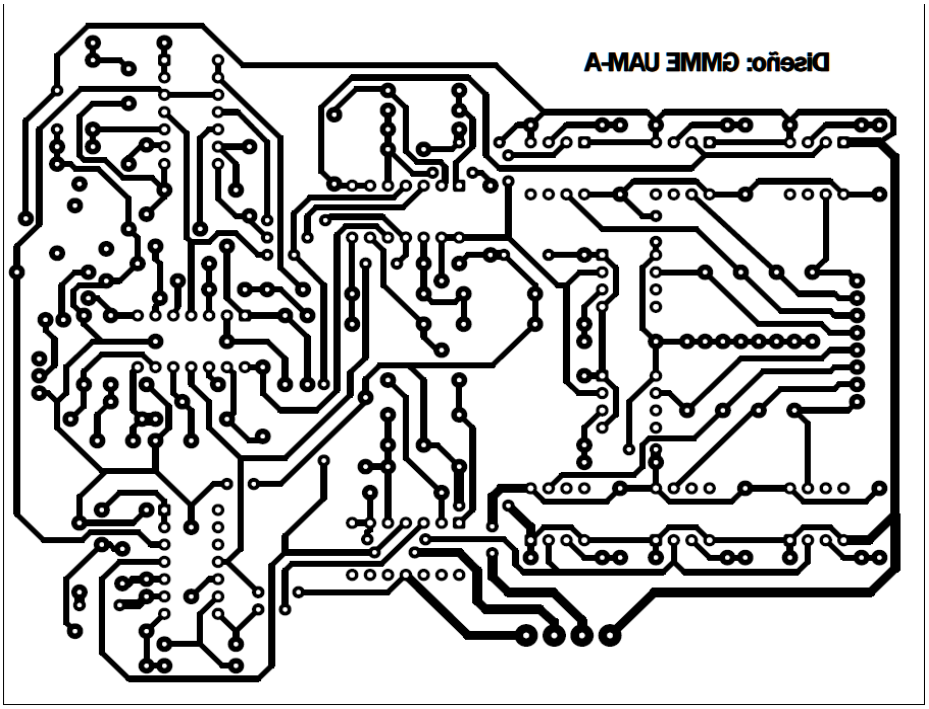
Circuito PCB del driver de los motores a pasos



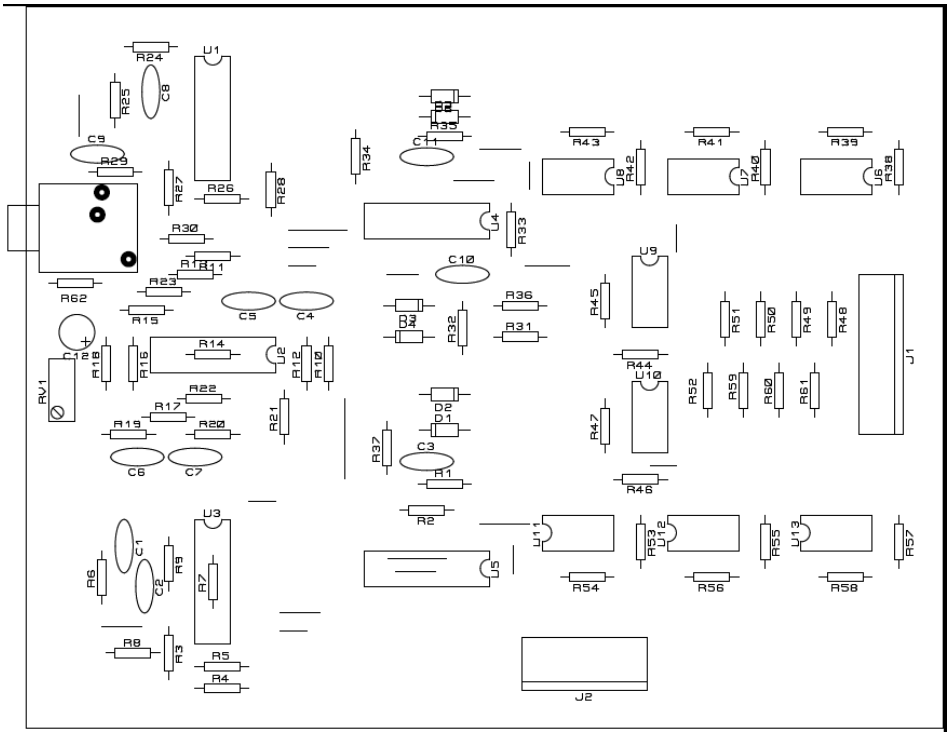
Mascara de componentes



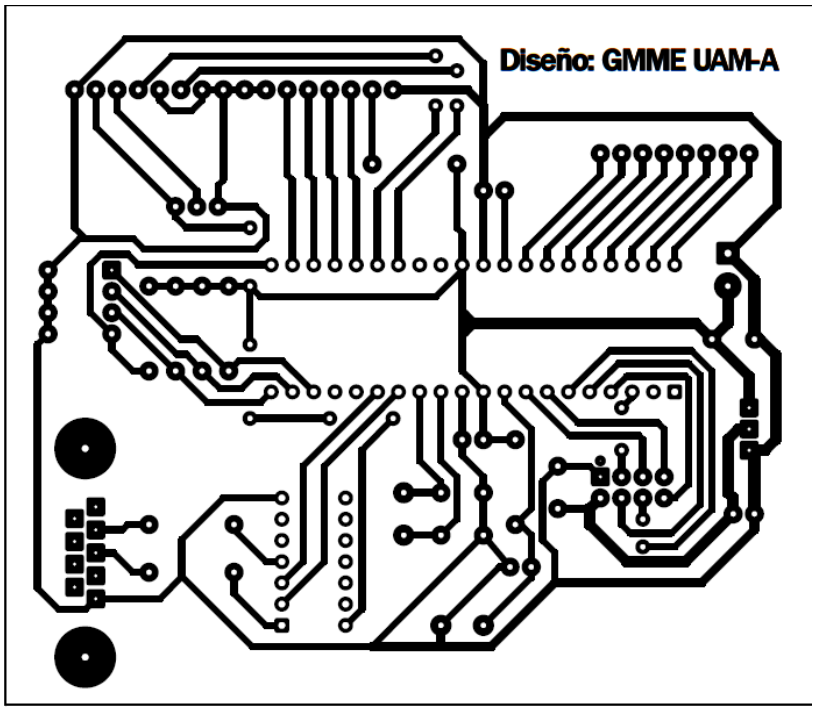
Circuito PCB de la etapa de procesamiento de audio



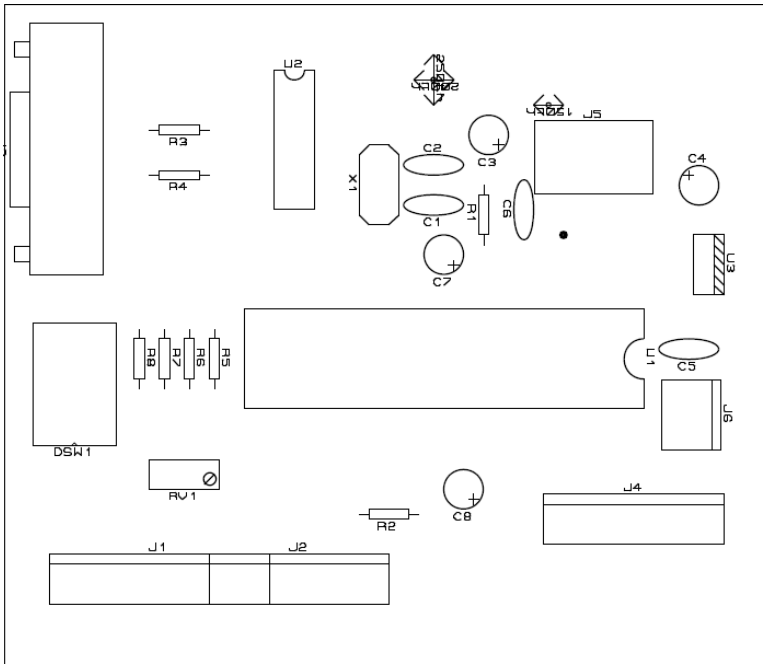
Mascara de componentes



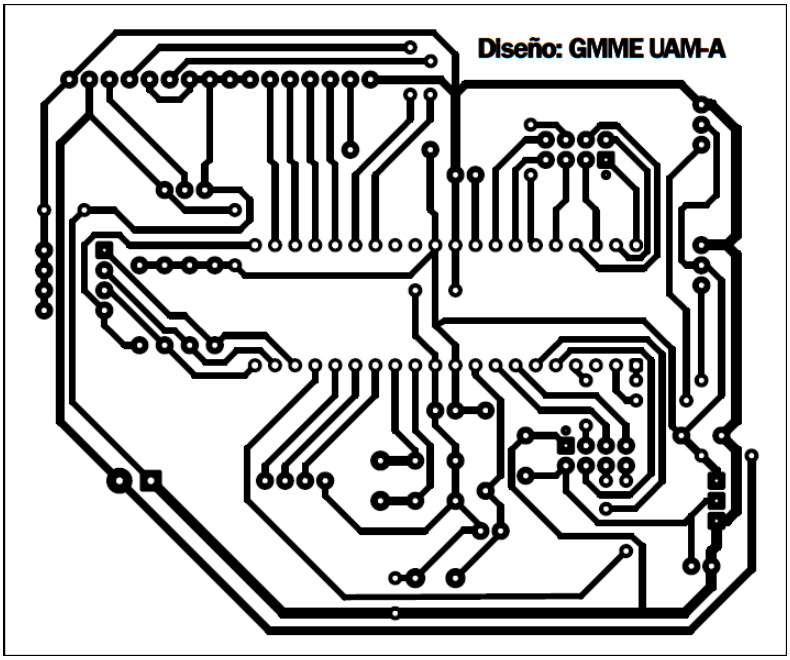
Circuito PCB del sistema maestro



Mascara de componentes



Circuito PCB del sistema esclavo



Mascara de componentes