Proyecto final

Diseño y fabricación de una pantalla POV (persistence of vision)

Realizado por:
Tintaya jose

Carrera:
Tecnicatura superior en electrónica

Ente educativo:
Instituto General Manuel Savio

Indice

Resumen

El efecto de Persistencia de la Visión es un fenómeno perceptivo clave que nos permite ver el mundo como una secuencia continua de imágenes, a pesar de que la información visual llega a nuestros ojos de forma discreta. Su comprensión y aplicación han dado lugar a numerosas tecnologías y formas de expresión artística, desde el cine hasta los modernos displays LED giratorios, demostrando el poder de la forma en que nuestro cerebro interpreta la fugaz información que captan nuestros ojos.

Este fenómeno óptico, inherente a la retina humana, provoca que una imagen luminosa persista en nuestra percepción por una fracción de segundo después de que la fuente de luz real haya desaparecido. Al explotar esta característica de la visión, es posible crear la ilusión de imágenes o texto "flotando" en el aire. La esencia de mi proyecto radica en la rotación de una tira de LEDs que se encienden y apagan en secuencias ultrarrápidas y sincronizadas. Cuando esta hélice gira a una velocidad suficiente, los destellos de luz individuales se fusionan en la mente del observador, formando una imagen coherente y aparentemente estática en el espacio.

Este proyecto de Persistencia de Visión no es solo un artefacto tecnológico; es una representación tangible de la formación integral recibida en la carrera de electrónica. Desde la concepción del diseño hasta la implementación práctica y la depuración, cada etapa ha permitido aplicar y consolidar los conocimientos teóricos.
Además de su función como demostrador básico del efecto POV, el sistema está diseñado con un potencial de expansión significativo. Futuras mejoras podrían incluir la capacidad de cargar imágenes personalizadas a través de una interfaz de comunicación (Bluetooth, Wi-Fi o tarjeta SD), la implementación de un control de velocidad de la hélice más sofisticado para adaptar dinámicamente la secuencia de LEDs, o incluso la adición de sensores para interactuar con el entorno.

En resumen, este proyecto no solo ilumina el fenómeno de la Persistencia de Visión, sino que también ilumina y valida cada hora de estudio y cada concepto aprendido en mi trayectoria como futuro ingeniero electrónico.

1- Introducción al Efecto POV (Persistencia de la Visión)

El **Efecto POV (Persistencia de la Visión)** es un fenómeno fascinante de la percepción visual humana que explota la forma en que nuestros ojos y cerebro procesan la información visual a lo largo del tiempo. En esencia, se basa en la capacidad del ojo humano para retener una imagen durante una fracción de segundo después de que la fuente original ha desaparecido. Esta "persistencia" de la imagen en nuestra retina es la clave para crear la ilusión de imágenes estáticas o en movimiento a partir de secuencias rápidas de luces o formas que aparecen y desaparecen.

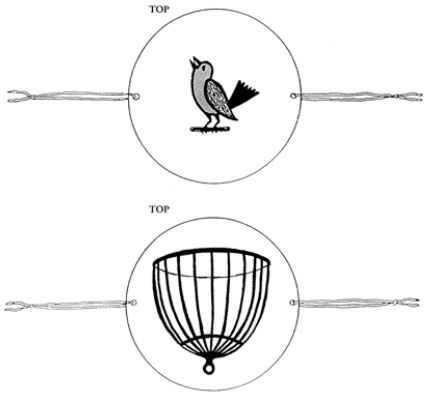


fig. 1.1 Taumatropo

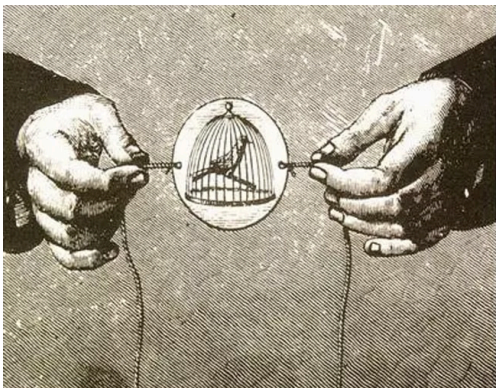


fig 1.2 Efecto optico taumatropo

1.2-Desglosando el fenómeno:

1. **La Retina y la Percepción Visual:** Cuando la luz incide en nuestros ojos, es enfocada por la córnea y el cristalino hacia la retina, la capa sensible a la luz en la parte posterior del ojo. La retina contiene millones de fotorreceptores: los conos (responsables de la visión del color y los detalles en condiciones de buena iluminación) y los bastones (más sensibles a la luz y responsables de la visión en condiciones de poca iluminación y la detección de movimiento). Estos fotorreceptores convierten la luz en señales eléctricas que son enviadas al cerebro a través del nervio óptico, donde se interpretan como imágenes.

2. **El Tiempo de Retención Visual:** La persistencia de la visión se debe a que los fotorreceptores de la retina no dejan de enviar señales eléctricas al cerebro inmediatamente cuando la luz desaparece. Existe un breve período de "decaimiento" en la actividad de estos receptores, durante el cual la impresión de la imagen original persiste en nuestra percepción. Se estima que esta persistencia visual dura aproximadamente entre **1/10 y 1/25 de segundo** (alrededor de 40 a 100 milisegundos).

3. **La Ilusión de Continuidad:** El efecto POV se aprovecha presentando una serie de imágenes o puntos de luz de forma secuencial ya una velocidad lo suficientemente alta como para que la imagen anterior aún esté "grabada" en nuestra retina cuando aparece la siguiente. Nuestro cerebro, al integrar estas imágenes fugaces que se superponen temporalmente, las percibe como una imagen continua, un texto estático

o incluso una animación fluida.

1.3 Cómo se manifiesta el efecto POV

El efecto POV se puede observar en diversas aplicaciones y fenómenos cotidianos:

- **El Cine y la Televisión:** La ilusión de movimiento en películas y videos se basa en la proyección rápida de una secuencia de fotogramas estáticos. La velocidad de proyección (generalmente 24 o 30 fotogramas por segundo) es lo suficientemente alta como para que la persistencia de la visión combine las imágenes sucesivas en un movimiento continuo.

- **Juguetes Ópticos:** Los juguetes como el **taumatropo** (un disco con dos imágenes diferentes en cada lado que al girar rápidamente se superponen), el **fenakistoscopio** (un disco giratorio con una serie de dibujos que al mirarse a través de ranuras da la ilusión de movimiento) y el **zoótropo** (un cilindro con dibujos en el interior que al girar y mirarse a través de ranuras crea una animación) son ejemplos clásicos que demuestran el principio de la persistencia de la visión.

- **Pantallas LED Giratorias (POV Displays):** Esta es una de las aplicaciones más directas y llamativas del efecto POV. Se utilizan filas de LED que se encienden y apagan rápidamente mientras se desplazan linealmente o rotan. Debido a la velocidad del movimiento y al rápido parpadeo de los LED, nuestro ojo integra los puntos de luz sucesivos en formas, letras, números o incluso animaciones completas que parecen flotar en el aire.

- **Estroboscopios:** Las luces estroboscópicas emiten destellos de luz a intervalos regulares. En ciertas frecuencias, pueden crear ilusiones visuales interesantes, como hacer que objetos en movimiento parezcan estáticos o moverse lentamente hacia atrás. Esto también está relacionado con la persistencia de la visión y la forma en que el cerebro interpreta las imágenes intermitentes.

- **Dibujo con Luces (Light Painting):** Al mover una fuente de luz en la oscuridad mientras se toma una fotografía con una exposición prolongada, la cámara captura la trayectoria de la luz como líneas continuas, creando formas y diseños. Aunque no es una ilusión directa para el ojo en tiempo real, el principio subyacente de capturar el movimiento de la luz a lo largo del tiempo está relacionado con la idea de la persistencia visual.

1.4 Implicaciones y Aplicaciones Tecnológicas

El efecto POV ha sido fundamental en el desarrollo de tecnologías de visualización y sigue siendo relevante en la actualidad:

- **Pantallas de Baja Resolución:** Permite crear la ilusión de imágenes de mayor resolución con un número limitado de elementos emisores de luz.

- **Interfaces Interactivas:** Se están explorando aplicaciones para crear interfaces tridimensionales o hologramas "falsos" utilizando pantallas POV interactivas.
- **Publicidad y Marketing:** Los displays POV son llamativos y pueden utilizarse para crear anuncios y mensajes visuales impactantes.
- **Arte y Entretenimiento:** Los artistas utilizan el efecto POV para crear instalaciones visuales innovadoras y experiencias interactivas.

2 Objetivo del proyecto

El objetivo principal de este proyecto académico es la fabricación de una pantalla giratoria basada en la técnica de Persistencia de la Visión (POV). A través del diseño e implementación de un sistema que controle la activación y desactivación secuencial de una hilera de LEDs en rotación, se busca demostrar cómo la persistencia retiniana del ojo humano permite percibir una imagen estática o en movimiento a partir de la rápida sucesión de puntos luminosos.

2.1 Objetivos específicos

- Diseñar el sistema mecánico para la rotación controlada de la hilera de LEDs.
- Seleccionar e implementar el circuito electrónico necesario para controlar individualmente el encendido y apagado de cada LED.
- Desarrollar el software o firmware que gestione la secuencia de activación de los LEDs en función de la imagen deseada y la velocidad de rotación.
- Demostrar experimentalmente cómo la velocidad de rotación y la frecuencia de actualización de los LED interactúan para crear la ilusión de una imagen estable debido al fenómeno de la persistencia de la visión.
- Analizar y documentar los resultados obtenidos, incluyendo las limitaciones y posibles mejoras del sistema desarrollado.

3 Prototipo

La creación de un prototipo de pantalla POV rotatoria con Arduino se considera por una combinación de factores que abarcan desde el aprendizaje y la exploración tecnológica hasta el potencial para aplicaciones prácticas y creativas:

- **Demostración y Validación del Principio POV:** Un prototipo tangible permite demostrar de manera visual y práctica el objetivo del proyecto. El prototipo sirve como una prueba de concepto clara y convincente.

- **Exploración de la Integración de Hardware y Software:** El proyecto involucra la integración de diversos componentes de hardware (motor, LEDs, sensor de posición opcional, Arduino) y software (programación del microcontrolador para control, sincronización y visualización de datos). Construir el prototipo ofrece una valiosa experiencia práctica en la interconexión, configuración y programación de estos elementos, habilidades fundamentales en robótica, electrónica y sistemas embebidos.

- **Aprendizaje Interdisciplinario:** La creación de la pantalla POV es un proyecto inherentemente interdisciplinario, que abarca conceptos de física (óptica, movimiento rotacional), electrónica (control de LEDs, sensores), programación (control de tiempo, lógica, algoritmos de visualización) y diseño mecánico (construcción de la hélice, montaje del motor). Esto lo convierte en una excelente herramienta de aprendizaje para estudiantes, aficionados y profesionales que buscan ampliar sus conocimientos en diversas áreas.

- **Potencial para la Innovación y la Creatividad:** Una vez que se comprende el funcionamiento básico, el prototipo se convierte en una plataforma para la experimentación y la innovación. Se pueden explorar diferentes diseños de hélices, patrones de LEDs, algoritmos de visualización más complejos (imágenes a color, animaciones), métodos de control interactivo (sensores de movimiento, comunicación inalámbrica) y aplicaciones artísticas o informativas novedosas.

- **Desarrollo de Habilidades Técnicas Específicas:** El proyecto fomenta el desarrollo de habilidades técnicas concretas como:

    - Programación en el entorno Arduino (C/C++).
    - Control de periféricos (motores, LEDs, sensores).
    - Gestión del tiempo y sincronización precisa.
    - Diseño y fabricación de prototipos electrónicos y mecánicos.
    - Solución de problemas y depuración de sistemas integrados.

- **Bajo Costo y Accesibilidad:** Arduino es una plataforma de prototipado de bajo costo y ampliamente accesible, lo que permite a individuos y grupos con presupuestos limitados explorar conceptos avanzados de electrónica y programación. Los componentes necesarios para una pantalla POV básica son relativamente económicos y fáciles de conseguir.

3.1 Proceso de elaboración del prototipo

Para el diseño se dividirá la construcción en 5 partes:

- **Estructura**

  Esta parte constituye de la infraestructura que va a sostener todo el dispositivo, ya que en un principio, los componentes están montados sobre una placa perforada y

para la conexión estará fijada por cables por lo que el módulo de proyección tendrá más peso y desbalance, por esto la base de la estructura debe ser más pesada.

El soporte en el que estará sujeto el motor debe tener 3 cm más de longitud que la mitad del módulo de proyección.

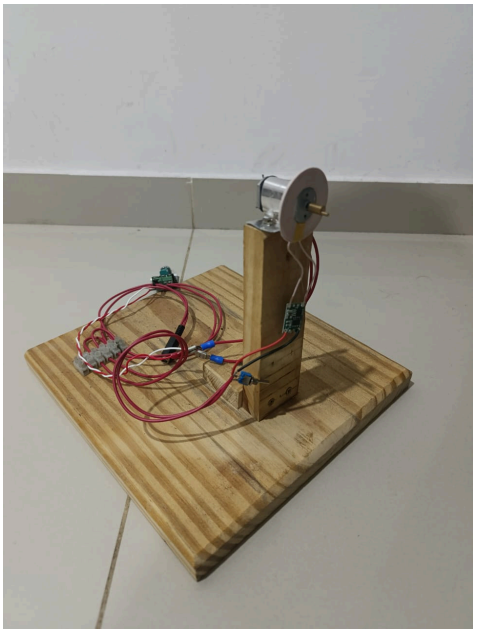


fig. 2.1 base de madera

Debido a su maleabilidad , peso y rigidez me pareció adecuado el uso de madera como base para el prototipo ya que también amortigua las vibraciones causadas por la rotación de la hélice.

- **Módulo de giro**

A este constituye todas las partes que harán posible que el dispositivo pueda realizar los giros necesarios para observar las imágenes en la pantalla pov, está compuesto por:

- ☐ Fuente de alimentación de 12v / 1A: para alimentar este proyecto es necesario pasar de 220v AC de la red domiciliaria a 12v / 1a DC, entre la variedad de fuentes disponibles, una gran gran opción es una switching, debido a su tamaño, costo bajo y la estabilidad necesaria para no tener problemas con el suministro eléctrico.

- ☐ motor dc 6v-12v: sobre el eje de este motor, estará montada la hélice de proyección cuyo peso (en la etapa de prototipo) será significativo, por lo que la potencia del motor debe ser un punto clave para la selección del mismo.

- ☐ Controlador de velocidad : 1 potenciómetro de 250k, 3 resistencias de 1k, 1 transistor mosfet irfz44n, 1 circuito integrado ne555 (más socket), 2 diodos 1n4148, 1 diodo led, 2 capacitores 1nf, 2 borneras de dos pines.

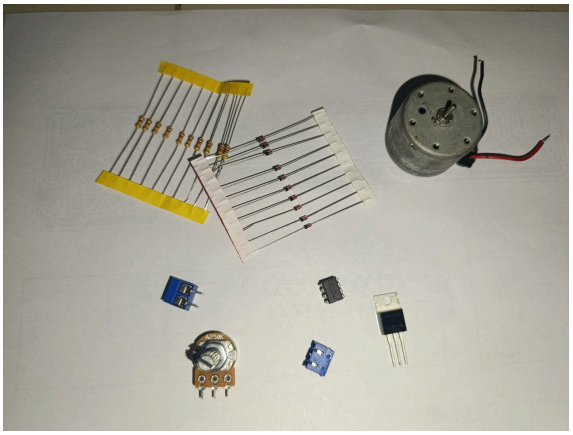


fig. 2.2 componentes electrónicos

- **Módulo de proyección**

  La hélice será la parte compuesta, que se encargará de presentar la secuencia de luces que en sincronía con el módulo de giro proyectan la imagen dada por el programa c++

  - ☐ Arduino nano: Es una herramienta increíblemente útil para proyectos electrónicos, especialmente aquellos de tamaño reducido o con requisitos de espacio limitados. Su principal fortaleza radica en ser una versión compacta del popular Arduino Uno, ofreciendo prácticamente la misma funcionalidad en un formato mucho más pequeño.

  - ☐ Leds 5mm comunes (11 unidades): las luces que desprenden estos diodos led, serán los que interactúen directamente con las retinas del espectador, dando lugar una imagen que nuestro cerebro interpreta como estática.

  - ☐ Resistencia 120 ohm (11 unidades): estos componentes limitan la corriente que llega a los diodos led, para evitar el deterioro de los mismos.

  - ☐ Sensor de efecto hall A3144: El A3144 es un sensor digital que detecta la presencia de un campo magnético . Para implementarlo, se fija un pequeño imán en un punto estratégico de la hélice giratoria (por ejemplo, en la base o cerca del eje de rotación). El sensor Hall se monta en una posición fija en el chasis del proyector, de manera que el imán pasa muy cerca de él en cada

revolución. Cuando el imán pasa por el sensor, este lo detecta y envía una señal digital (un pulso alto o bajo) al microcontrolador. Este pulso actúa como un "marcador" o "gatillo". El microcontrolador utiliza esta señal para determinar el inicio de cada revolución Calcular la velocidad de rotación y sincronizar la secuencia de LEDs.

- ☐ Cables 28 AWG: Para la comunicación entre los diferentes componentes y su respectiva alimentación, en este prototipo utilizaremos cables del menor diámetro posible.
- ☐ Placa perforada: Para facilitar el posicionamiento de los componentes, se optó por usar una placa perforada.
- ☐ Bobina de alimentación inalámbrica (receptor): Uno de los puntos a tener en cuenta, al momento de alimentar todo el módulo de giro, fue el peso. Una batería (dependiendo la capacidad) añade volumen y peso al modulo, unos anillos colectores dificultan el acople al motor dc además de agregar costo al presupuesto final, para su autonomía y mejor integración al módulo, se llegó a la conclusión de una alimentación por inducción es la opción más óptima

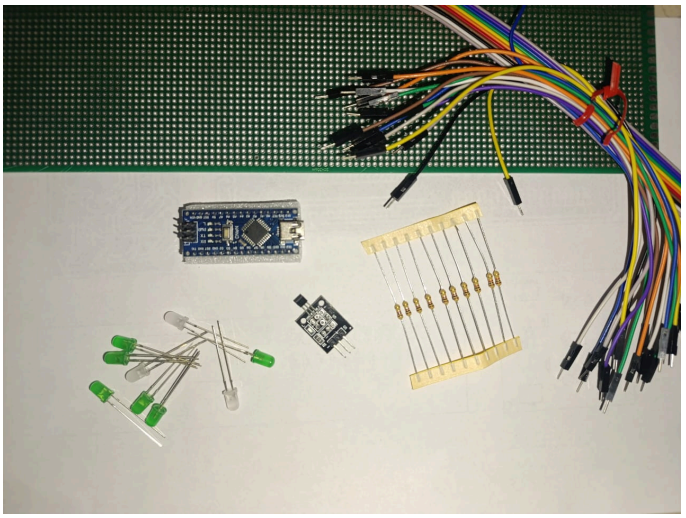


fig.2.3 componentes electrónicos

- **Programación de la lógica de todo el circuito**

  Codigo en arduino en lenguaje c++.

- **Módulos de alimentación**

  Fuente de alimentación

  Bobinas alimentación inalámbrica (emisor)

fig. 2.4 módulo inalámbrico fig. 2.5 fuente de alimentación

3.2 Funcionamiento de la pantalla pov

El corazón de este dispositivo radica en un fenómeno de la percepción visual humana llamado **persistencia de la visión**. Nuestros ojos y cerebro no procesan la luz de forma instantánea. Cuando una imagen desaparece de nuestro campo de visión, la retina retiene una "huella" visual por una fracción de segundo (aproximadamente 1/16 a 1/10 de segundo). Si una nueva imagen similar o complementaria se presenta en ese breve lapso, nuestro cerebro tiende a fusionar ambas, creando la ilusión de una imagen continua.

En el contexto de esta pantalla rotatoria, los LEDs que se encienden y apagan rápidamente en diferentes posiciones durante la rotación de la hélice aprovechan esta persistencia para "dibujar" la imagen completa en el aire a medida que nuestro cerebro integra las sucesivas líneas de luz.

3.3 Componentes Clave y su Interacción

A. **La Helice**

a. Es la estructura física que soporta los LEDs y el circuito. por lo que el material debe ser ligero para facilitar la rotación con un motor pequeño.
b. Su movimiento rotatorio es crucial para "barrer" el espacio y presentar los LEDs en diferentes posiciones angulares rápidamente.

B. **El Motor**

a. Proporciona la fuerza para hacer girar la hélice a una velocidad constante y controlada.

b. La velocidad de rotación es un parámetro crítico. Debe ser lo suficientemente rápida para que la persistencia de la visión funcione de manera efectiva y la imagen parezca sólida, pero no tan rápida que cause desenfoque o dificultades para sincronizar los LEDs.

.

C. **El Circuito de LEDs**

a. Una serie de LEDs (generalmente monocromáticos o RGB) están estratégicamente colocados a lo largo de la hélice. La disposición y la cantidad de LEDs determinan la resolución vertical de la imagen proyectada.
b. Cada LED puede ser encendido o apagado individualmente por el microprocesador.

D. **El Microprocesador Arduino**

a. Es el "cerebro" del sistema. Su función principal es **controlar el encendido y apagado de cada LED en el momento preciso** durante la rotación de la hélice.
b. Para lograr esto, el Arduino debe:
    i. **Conocer el ángulo actual de la hélice:** Esto se puede lograr mediante el uso de un sensor de posición rotatoria (encoder rotativo) acoplado al motor o a la hélice. El encoder envía pulsos al Arduino a medida que gira, permitiéndole determinar el ángulo exacto en cada instante.
    ii. **Conocer la velocidad de giro:** Monitoreando la frecuencia de los pulsos del encoder, el Arduino puede calcular la velocidad de rotación. Mantener una velocidad constante es importante para una imagen estable.
    iii. **Almacenar la información de la imagen/texto:** El Arduino tiene en su memoria (o puede recibirla externamente) la secuencia de bits que representa la imagen o el texto que se va a proyectar. Esta información se organiza generalmente en "columnas" o "franjas" verticales de píxeles, correspondientes a cada posición angular de la hélice.
    iv. **Sincronizar el encendido de los LEDs:** Basándose en el ángulo actual, la velocidad de giro y la información de la imagen, el Arduino activa o desactiva cada LED individualmente en el momento exacto en que pasa por una determinada posición angular.

3.4 El Proceso de Proyección de la Imagen/Texto

A. **Detección de la Posición Angular:** A medida que el motor gira la hélice, el sensor de posición (si lo hay) informa continuamente al Arduino sobre el ángulo actual. Si no hay un sensor dedicado, el Arduino puede estimar la posición basándose en el tiempo transcurrido y la velocidad del motor (aunque esto es menos preciso).

B. **Determinación de la "Columna" a Mostrar:** Para cada pequeña porción del ángulo de rotación, el Arduino determina qué columna vertical de píxeles de la imagen o qué parte de la letra debe mostrarse. Imagina la imagen final dividida en muchas franjas verticales del ancho de un LED.

C. **Activación Selectiva de los LEDs:** En el instante en que la hélice alcanza el ángulo correspondiente a una columna específica, el Arduino consulta la información almacenada para esa columna. Para cada LED en la hélice, el Arduino decide si debe encenderse o apagarse según el color (si son RGB) o la intensidad requerida para ese píxel en esa posición vertical.

D. **Barrido Rápido:** Este proceso de encender y apagar los LEDs según la columna de información se repite muy rápidamente a medida que la hélice completa una rotación. Cada LED "dibuja" una línea vertical de la imagen en el aire.

E. **Integración por el Ojo Humano:** Debido a la velocidad de rotación y la persistencia de la visión, nuestro cerebro no percibe los LEDs individuales encendiéndose y apagándose en diferentes posiciones. En cambio, integra todas estas líneas de luz sucesivas, creando la ilusión de una imagen o texto estático y completo flotando en el espacio.

3.5 Factores Críticos para una Buena Proyección

- **Velocidad de Rotación Constante:** Las variaciones en la velocidad de giro pueden causar distorsiones en la imagen (estiramiento o compresión). Un control preciso del motor es esencial.
- **Precisión de la Sincronización:** La sincronización entre el ángulo de la hélice y el momento en que se encienden los LEDs debe ser muy precisa. Cualquier desfase puede resultar en imágenes borrosas o desalineadas.
- **Resolución (Número de LEDs):** Cuantos más LEDs haya en la hélice, mayor será la resolución vertical de la imagen proyectada.
- **Frecuencia de Actualización:** La velocidad de rotación determina la frecuencia con la que se "refresca" la imagen completa. Una frecuencia demasiado baja puede provocar parpadeo.

3.6 Problemas emergentes

3.6.1 Sobrecalentamiento en el módulo de carga inalámbrica

A. Para el Emisor (Alimentado con 12V):

- **Verifica la Corriente de Entrada:** Asegurarse de que la fuente de alimentación de 12V esté proporcionando la corriente adecuada que requiere el emisor. Una fuente de alimentación con una capacidad de corriente insuficiente podría forzar al emisor y generar calor. Revise las especificaciones del módulo.
- **Disipación de calor:**
    - **Asegurarse de que haya suficiente espacio alrededor del emisor para la circulación de aire.** No lo obstruyas con otros objetos.
    - **Considerar en agregar un pequeño disipador de calor al chip principal del emisor.** Se puede encontrar fácilmente en tiendas de electrónica. Asegurarse de usar pasta térmica entre el chip y el disipador para una mejor transferencia de calor.
    - **En casos más extremos, podría considerarse en un pequeño ventilador** que dirija aire hacia el emisor.
- **Eficiencia del Circuito:** Si se tiene acceso al circuito del emisor, verificar si hay componentes que se ven visiblemente dañados o que están funcionando de manera ineficiente, como reguladores de voltaje o transistores.
- **Calidad del módulo:** Si este es un módulo genérico de baja calidad, podría ser internamente menos eficiente y generar más calor. Considere probar con un módulo de un fabricante más reconocido.

B. Para el Receptor (Entrega 5V):

- **Carga del dispositivo:** Asegurarse de que el dispositivo que esté cargando no esté demandando más corriente de la que el receptor pueda manejar de manera eficiente. Algunos dispositivos pueden tener picos de consumo durante la carga.
- **Ubicación y contacto:**
    - **Asegurarse de que el dispositivo receptor esté correctamente alineado con el emisor.** Una mala alineación puede reducir la eficiencia de la transferencia de energía y generar más calor.
    - **Evitar colocar objetos entre el emisor y el receptor** , ya que esto puede interferir con la transferencia de energía y aumentar la temperatura.
- **Carcasa del dispositivo:** Si el receptor está integrado en una carcasa, asegurarse de que no esté atrapando el calor. Algunas carcasas gruesas o de materiales aislantes pueden empeorar el problema.
- **Disipación de Calor (Similar al Emisor):** Si es posible acceder al chip principal del receptor, considere agregar un pequeño disipador de calor.
- **Eficiencia del Circuito (Similar al Emisor):** Verificar si hay componentes defectuosos en el receptor.

C. Consideraciones Generales para Ambos:

- **Distancia entre Emisor y Receptor:** Intentar mantener la distancia entre el emisor y el receptor dentro del rango óptimo especificado por el fabricante. Una distancia excesiva puede reducir la eficiencia y generar más calor.
- **Entorno:** Evitar utilizar el cargador inalámbrico en ambientes con temperaturas elevadas o bajo la luz directa del sol.
- **Pruebas Individuales:** Probar el emisor y el receptor con diferentes dispositivos o fuentes de alimentación (si es posible y seguro) para aislar si el problema está en alguno de los componentes en particular.

Es importante tener en cuenta que **cierto nivel de calentamiento es normal** durante la carga inalámbrica debido a las pérdidas de energía en forma de calor. Sin embargo, si el sobrecalentamiento es excesivo al punto de que los componentes están muy calientes al tacto o incluso huelen a quemado, es una señal de que algo no está funcionando correctamente y debe tomarse para evitar daños mayores o riesgos de seguridad.

4 Esquematico de circuitos

4.1Esquemático del módulo de proyección

4.1.1 Tensión de entrada:

El módulo de alimentación por inducción está representado por la batería de 9v, en los pines de J1 estarán conectadas a la bobina.

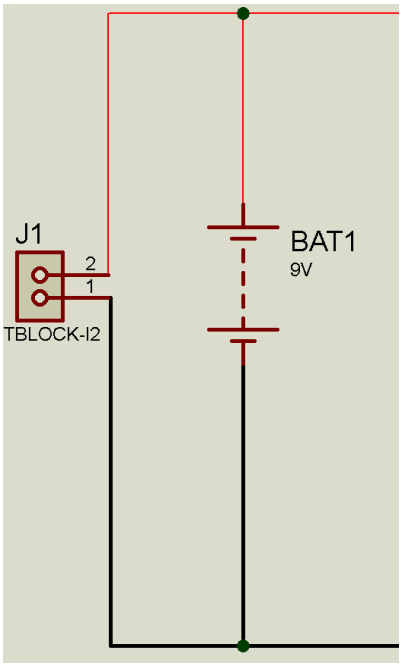


fig. 3.1 Esquemático módulo de giro

4.1.2 Sensor:

Para crear la interrupción, se conecta los pines del a3144 según lo indique el datasheet, después de alimentarlo (tensión de alimentación en pin 5V, neutro en pin GND), la salida

digital que proporciona se comunicara con el pin d13 del arduino nano, de este modo, el sensor creará la interrupción que el programa necesita para iniciar la secuencia de luces con el posicionamiento correcto.
Este modelo de sensor hall manda una señal de interrupción a partir de una excitación externa dada por un campo magnético.

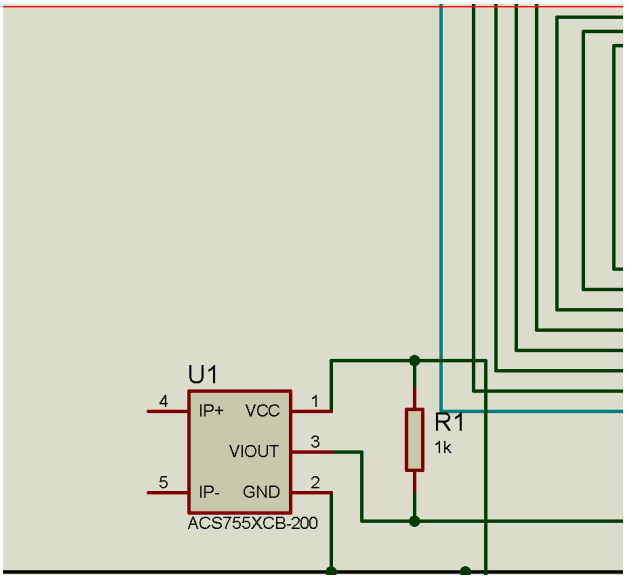


fig. 3.2 Esquemático módulo de giro

4.1.3 Microcontrolador

Una vez alimentado el arduino nano (conectando en VIN a la tensión de alimentación y en complemento también anexar el neutro a GND), se deben asignar las 11 salidas que darán lugar a la secuencia de luces led, por lo que en estas salidas serán las encargadas de enviar las señales de designe el programa dentro del microcontrolador, en este caso se eligieron los pines del d2 al d12.

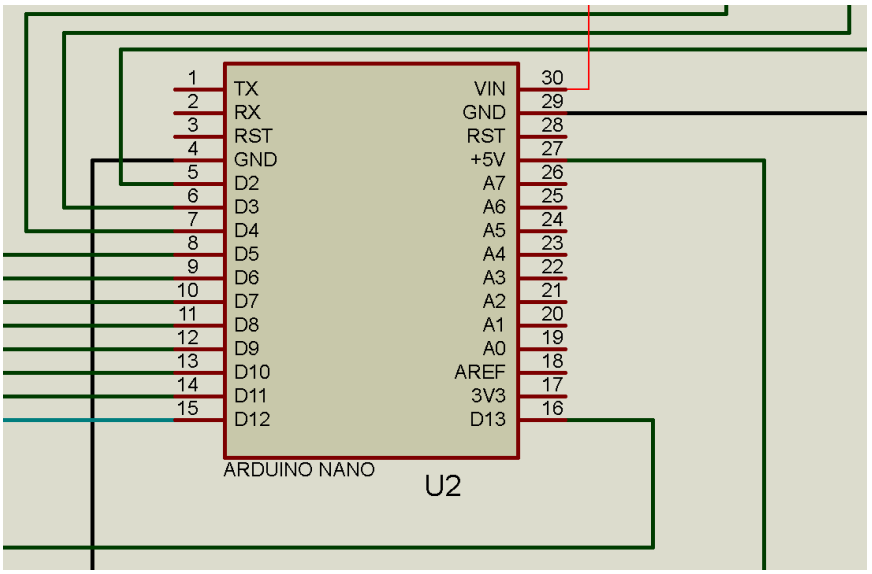


fig. 3.3 Esquemático módulo de giro

4.1.4 Barra led:

Ya teniendo configurada las salidas y entradas del arduino nano, ahora comenzamos con el conexionado de los led, cada led se estara derribando de una salida del arduino del d2 al d12, vinculando cada salida con el anodo de los diodos pondremos una resistencia limitadora de corriente.

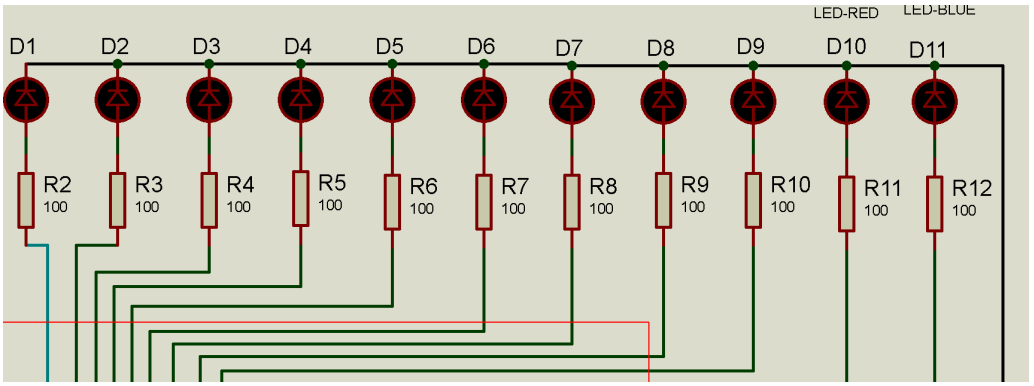


fig. 3.4 Esquemático módulo de giro

4.2 Esquemático control de velocidad

4.2.1 Alimentación:

Como V input, se cuenta con la alimentación que entrega la fuente switching, es la fuente de alimentación principal del circuito. Proporciona una tensión constante de 12 voltios (CC) a todo el sistema.

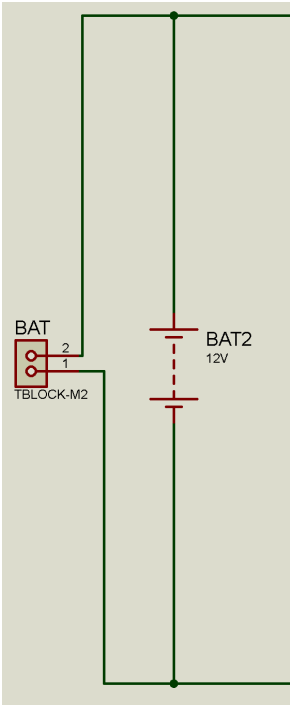


fig. 4.1 esquemático de control de velocidad

4.2.2 Entrada de corriente:

D3 (LED-amarillo) el diodo emisor de luz de color amarillo, actúa como un indicador visual de que el circuito está encendido o funcionando. R1 (3,3 kΩ) Una resistencia limitadora de corriente para el LED D3. Evita que una corriente excesiva fluya a través del LED y lo dañe. Calcula la corriente I=V / R, donde V es la caída de tensión sobre la resistencia (aproximadamente 12 voltios−2 voltios del LED), y Res 3,3 kΩ.

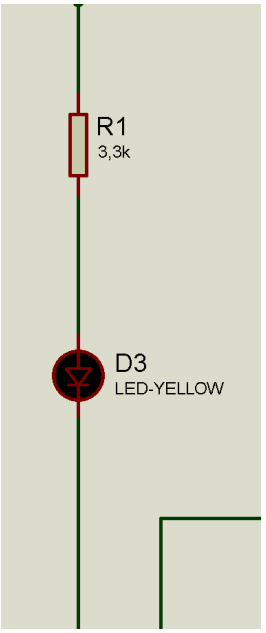


fig. 4.2esquemático de control de velocidad

4.2.3 Integrado 555 :

U4 (Temporizador 555): Siendo este, el corazón del circuito, el integrado temporizador 555. Está configurado en modo **astable** , lo que significa que genera una **onda cuadrada continua** (pulsos repetitivos) en su pin de salida (Pin 3). La frecuencia y el ciclo de trabajo de esta onda se determinan por los valores de R2, R3, RV4, D1, D2 y C1.

Pines Clave del 555 en esta configuración: Para una correcta disposición de los pines, se tiene como referencia la datasheet del integrado.

- **Pin 8 (VCC):** Conectado a la alimentación positiva (12V).
- **Pin 1 (GND):** Conectado a tierra.
- **Pin 4 (Reset):** Conectado a VCC para asegurar que el temporizador siempre esté habilitado.
- **Pin 5 (Voltaje de control - CV):** Conectado a tierra a través de C2 (10nF). Este capacitor se usa para filtrar ruido o para modular externamente el ciclo de trabajo, aunque aquí solo es un filtro.
- **Pin 7 (Descarga - DIS):** Este pin es un transistor interno que descarga el condensador C1.
- **Pin 6 (Umbral - TH):** Monitorea la tensión en el capacitor C1. Cuando C1 se carga a2/3de VCC, el temporizador cambia de estado.

- **Pin 2 (Trigger - TR):** Monitorea la tensión en el capacitor C1. Cuando C1 se descarga a1/3de VCC, el temporizador cambia de estado.
- **Pin 3 (Output - OUT):** La salida de la onda cuadrada generada por el 555.

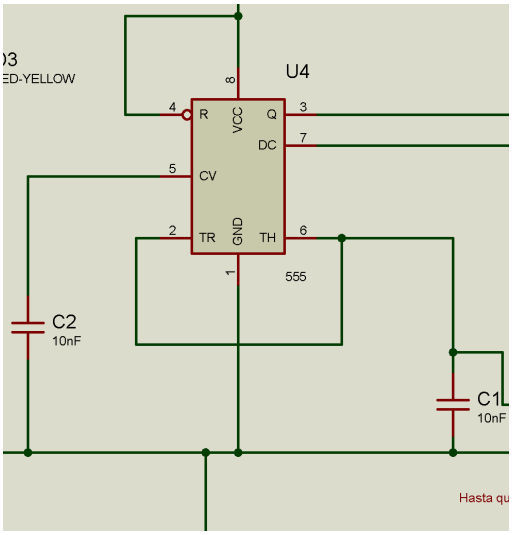


fig. 4.3 esquemático de control de velocidad

4.2.4 Regulador de potencia:

RV4 (250kΩ Potenciómetro) y R3 (1kΩ): El potenciómetro RV4 y la resistencia R3 controlan la ruta de descarga de C1. La combinación de R3 y RV4 permite ajustar la resistencia total de descarga. este potenciómetro permite variar la frecuencia de la onda cuadrada:

288 Hz a máxima resistencia: Cuando RV4 está en su valor máximo (250kΩ), la resistencia de descarga es mayor, la descarga de C1 es más lenta, resultando en una menor frecuencia (período más largo).

100 kHz a resistencia mínima: Cuando RV4 está en su valor mínimo (0Ω), la resistencia de descarga es menor, la descarga de C1 es más rápida, resultando en una mayor frecuencia (período más corto).

D1 (1N4148) y D2 (1N4148): Diodos de señal. Estos diodos son cruciales para separar los caminos de carga y descarga de C1, permitiendo un control más preciso sobre el ciclo de trabajo (el tiempo que la salida está alta vs. baja).

D1: Permite que C1 se cargue a través de R2.

D2: Permite que C1 se descargue a través de R3 y RV4 cuando el Pin 7 (Discharge) del 555 esté activo (a tierra).

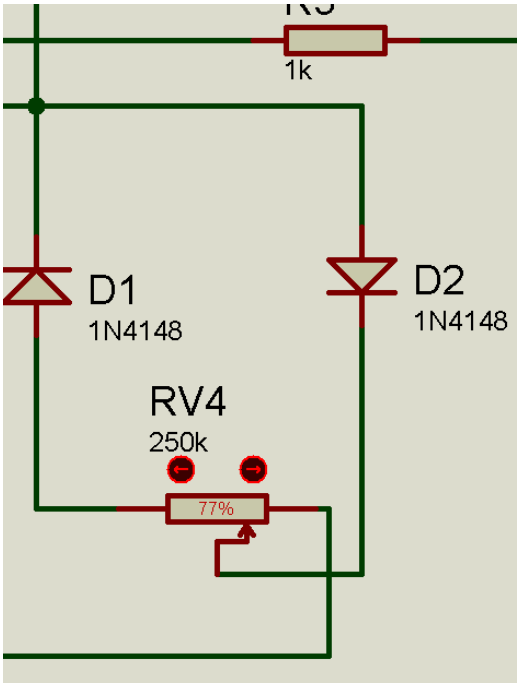


fig. 4.4 esquemático de control de velocidad

4.2.5 Caudal de potencia con mosfet

Q1 (IRFZ44N - MOSFET de canal N): Función: Es un **transistor MOSFET de canal N** de potencia. Actúa como un **interruptor controlado electrónicamente**. La señal de onda cuadrada del Pin 3 del 555 se aplica a su terminal de Gate (G). Cuando la salida del 555 es **alta (aprox. 12V)**, el Gate del MOSFET se polariza, y este se **activa (enciende)**, permitiendo que la corriente fluya entre su Drain (D) y su Source (S). Cuando la salida del 555 es **baja (aprox. 0V)**, la puerta del MOSFET no está polarizada, y este se **desactiva (apaga)**, interrumpiendo el flujo de corriente entre Drain y Source.

R2 (1kΩ): Resistencia a través de la cual C1 se carga cuando la salida del 555 es alta.

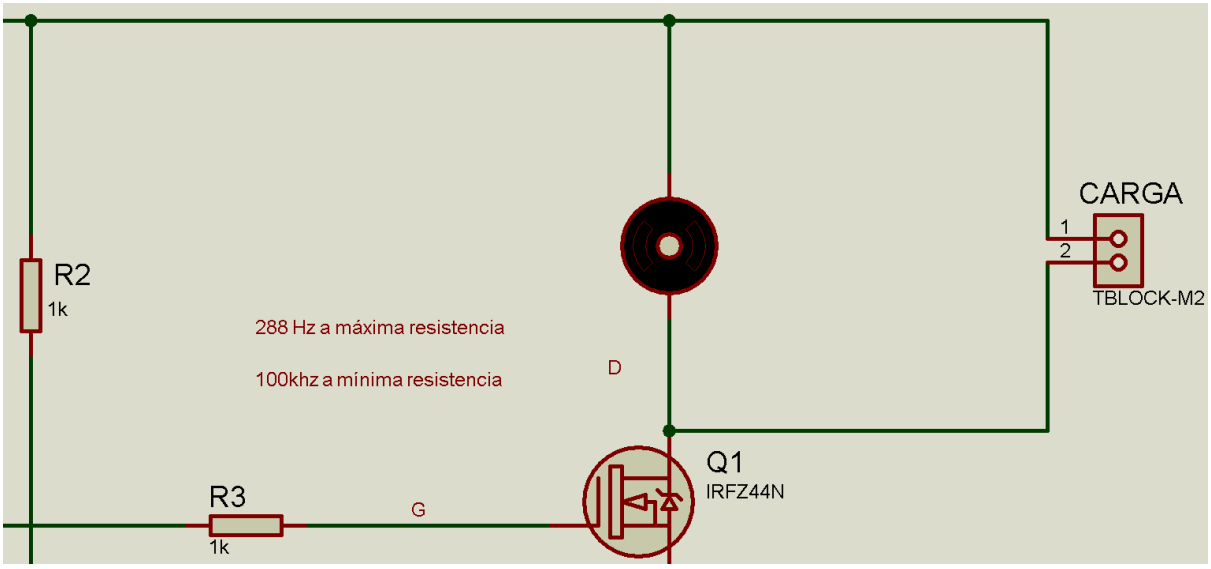


fig. 4.5 esquemático de control de velocidad

4.3 Configuración y cálculo:

4.3.1 Hélice (módulo de proyección):

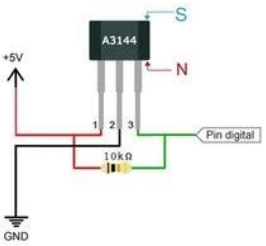


fig. 5.1 fig. 5.2 fig. 5.3

led verde:

- Vled = 3.2V (aproximadamente)
- Iled = 0.02A

R = (Vfuente − Vled) / Iled

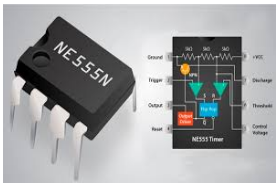
R=(5V − 3.2V) / 0.02A = 1.8V / 0.02A = 90Ω

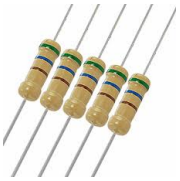
resistencia comercial 100Ω

Iled = (5V − 3.2V) / 100Ω = 1.8V / 100Ω = 0.018A = 18mA

Rhall = entre 1kΩ y 10kΩ (segun el fabricante)

4.3.2 Multivibrador Astable 555





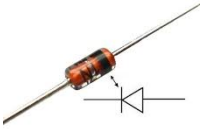


fig. 6.1 n555 fig. 6.2 resist. ig. 6.3 capac. fig. 6.4 diodo fig. 6.5 potenc. fig. 6.6 led

fig. 6.7 mosfet

Valores de los Componentes:

- Vsuministro=12V
- R1=3.3kΩ (Limitador de corriente para el LED D3)
- R2=1kΩ (Resistencia de temporización)
- R3=1kΩ (Resistencia de temporización)
- RV4=250kΩ (Potenciómetro, para control de frecuencia y ciclo de trabajo)
- C1=10nF (Condensador de temporización)
- C2=10nF (Condensador de desacoplo para el 555)
- D1,D2=1N4148 (Diodos de pequeña señal, para dirigir la corriente de carga/descarga)
- Q1=IRFZ44N (MOSFET de canal N, elemento de conmutación para el motor)
- D3 = LED-AMARILLO (LED indicador)
- Motor: Motor de corriente continua de 12V (CARGA)

- Voltaje de Operación: 12V CC
- Rango de Frecuencia PWM: 288 Hz a 100 kHz (controlado por RV4)
- Control del Ciclo de Trabajo: Rango completo (0-100%) para la variación de la velocidad del motor (controlado por RV4)
- Corriente Máxima del LED: Aproximadamente 3.03 mA

Ecuaciones para el Temporizador 555 en Modo Astable con Diodos

Funcionamiento del 555 en esta configuración:

- Periodo de Carga (tHIGH o Ton): El capacitor C1 se carga desde VCC a través de R2 y el diodo D1. El pin 7 (Discharge) del 555 está "apagado" (alta impedancia). El ciclo de carga va desde 1/3 VCC hasta 2/3 VCC.

- Periodo de Descarga (tLOW o Toff): El capacitor C1 se descarga desde 2/3 VCC hasta 1/3 VCC a través de la resistencia R3, el potenciómetro RV4 y el diodo D2, hacia el pin 7 (Discharge) del 555, que está activo (conectado a tierra).

Valores de componentes:

- R2=1kΩ
- R3=1kΩ
- RV4=0Ω a 250kΩ
- C1=10nF=10×10−9F
- Se asume una caída de tensión en los diodos VD≈0.7V (para 1N4148, aunque para señal es a veces menor, 0.6V es una buena aproximación.

a. Periodo de Salida Alta (tHIGH o Ton)

Cuando la salida (pin 3) es alta, el capacitor C1 se carga a través de R2 y el diodo D1. La ecuación para el tiempo en que la salida es alta es:

$tHIGH=0.693\times R2\times C1$

- El factor 0.693 proviene del logaritmo natural de 2 (ln(2)), que es el tiempo que tarda un capacitor en cargar de 1/3 a 2/3 de la tensión de la fuente en un circuito RC.

Sustituyendo los valores: $R2=1k\Omega=1000\Omega$ $C1=10nF=10\times10-9F$

$tHIGH=0.693\times1000\Omega\times10\times10-9F$ $tHIGH=0.693\times10-5s$ $tHIGH=6.93\mu s$

b. Periodo de Salida Baja (tLOW o Toff)

Cuando la salida (pin 3) es baja, el capacitor C1 se descarga a través de R3, el potenciómetro RV4 y el diodo D2, hacia el pin 7 (Discharge) del 555.

La ecuación para el tiempo en que la salida es baja es:

$tLOW=0.693\times(R3+RV4)\times C1$

- Aquí, RV4 es el valor de resistencia ajustado del potenciómetro, que puede variar de 0Ω a 250kΩ.

Sustituyendo los valores: $R3=1k\Omega=1000\Omega$ $C1=10nF=10\times10-9F$

- Cuando RV4 es mínimo (0Ω): $tLOW\_min=0.693\times(1000\Omega+0\Omega)\times10\times10-9F$ $tLOW\_min=0.693\times1000\Omega\times10\times10-9F$ $tLOW\_min=0.693\times10-5s$ $tLOW\_min=6.93\mu s$

- Cuando RV4 es máximo (250kΩ=250000Ω): $tLOW\_max=0.693\times(1000\Omega+250000\Omega)\times10\times10-9F$ $tLOW\_max=0.693\times251000\Omega\times10\times10-9F$ $tLOW\_max=0.693\times2.51\times10-3s$ $tLOW\_max\approx1738.83\mu s$ $tLOW\_max\approx1.739ms$

c. Periodo Total (T)

El periodo total de la onda cuadrada es la suma de los tiempos alta y baja:

$T = tHIGH + tLOW$

- Periodo mínimo (cuando RV4 es mínimo): $Tmin=6.93\mu s+6.93\mu s=13.86\mu s$

- Periodo máximo (cuando RV4 es máximo): $Tmax=6.93\mu s+1738.83\mu s\approx1745.76\mu s\approx1.746ms$

d. Frecuencia (f)

La frecuencia es el inverso del periodo total:

f=1/T

- Frecuencia máxima (cuando RV4 es mínimo): fmax=1/Tmin=1/(13.86×10−6s) fmax≈72150.07Hz fmax≈72.15kHz

- Frecuencia mínima (cuando RV4 es máximo): fmin=1/Tmax=1/(1.74576×10−3s) fmin≈572.82Hz fmin≈572.82Hz

e. Ciclo de Trabajo (Duty Cycle - DC)

El ciclo de trabajo es la relación entre el tiempo en que la salida es alta y el periodo total, expresado como porcentaje:

DC=(tHIGH/T)×100%

- **Cuando RV4 es mínimo (0Ω):** DCmin=(6.93µs/13.86µs)×100%=0.5×100%=50%

- **Cuando RV4 es máximo (250kΩ):** DCmax=(6.93µs/1745.76µs)×100% DCmax≈0.003969×100%≈0.397%

Como se puede ver, con esta configuración de diodos, el tHIGH es fijo (determinada solo por R2 y C1), mientras que el tLOW varía significativamente con RV4. Esto significa que el ciclo de trabajo puede ser muy bajo, pero nunca puede ser superior a 50% (idealmente, si R2 y R3+RV4 pudieran ser iguales en algún punto, lo cual no sucede aquí).

Cálculos con Caída de Tensión del Diodo (Más preciso, pero a menudo se ignora para el 555).

Para más precisión y considerar la caída de tensión VD del diodo (aprox. 0.7V para 1N4148):

A. Periodo de Salida Alta (tHIGH): El capacitor se carga desde 1/3VCC hasta 2/3VCC a través de R2 y D1. La tensión efectiva de carga es VCC−VD.

   tHIGH=R2×C1×ln(VCC−VD−32VCCVCC−VD−31VCC)
   tHIGH=R2×C1×ln(31VCC−VD32VCC−VD)

   Para VCC=12V y VD=0.7V:
   tHIGH=R2×C1×ln(4V−0.7V8V−0.7V)=R2×C1×ln(3.37.3)=R2×C1×ln(2.212)
   tHIGH≈R2×C1×0.794

tHIGH≈0.794×1000Ω×10×10−9F=7.94μs (Un poco más alto que los 6.93μs del cálculo idealizado).

B. Periodo de Salida Baja (tLOW): El capacitor se descarga a través de (R3+RV4) y D2 hacia el pin 7 (que está a tierra). Aquí la caída del diodo tiene menos impacto directo en la ecuación logarítmica de descarga, ya que la descarga ocurre hacia una referencia de voltaje fija (tierra a través del pin 7).

tLOW=(R3+RV4)×C1×ln(2) tLOW=0.693×(R3+RV4)×C1 (Esta ecuación sigue siendo la misma que la idealizada, ya que la descarga es del 2/3VCC al 1/3VCC hacia tierra a través de las resistencias y el diodo).

5. Conclusión

La culminación de este proyecto final, pantalla de Persistencia de Visión (POV), no solo representa el cierre de una etapa académica significativa, sino que también encapsula la pasión y el aprendizaje que ha brindado la carrera de tecnicatura superior en Electrónica.

Este proyecto en particular fue una oportunidad inmejorable para plasmar de manera práctica todo lo aprendido a lo largo de la carrera. Desde el diseño conceptual hasta la implementación física, cada subsistema del proyector POV ,el control preciso del motor, la compleja secuencia de encendido y apagado de los LEDs, la detección de posición angular y la gestión energética, requirió la aplicación de principios fundamentales de la electrónica analógica, digital y de potencia. Fue gratificante ver cómo conceptos teóricos cobran vida y funcionalidad en un dispositivo que genera una ilusión visual tan llamativa.

Sin embargo, el camino no estuvo exento de desafíos. Donde se presentaron dificultades considerables fue la programación en lenguaje C para el microcontrolador. La necesidad de optimizar el código para el control en tiempo real de los LEDs y la sincronización con el sensor de efecto Hall demandó una comprensión de la arquitectura del microcontrolador, la gestión de interrupciones y la manipulación eficiente de bits. Si bien fue un proceso arduo de depuración y reescritura, proporcionó una valiosa experiencia en firmware. De igual manera, la construcción física del proyecto presentó sus propios obstáculos, desde la precisión en el montaje de la hélice para evitar vibraciones excesivas hasta el diseño y

enrutamiento del cableado en un espacio reducido. Estas dificultades demandaron pensamiento creativo y a aplicar soluciones de ingeniería práctica.

A pesar de las complejidades, este proyecto, permitió reforzar y consolidar conocimientos esenciales que se consideran pilares de nuestra profesión. La matemática fue crucial para calcular las frecuencias de conmutación, los tiempos de carga y descarga de capacitores, y las relaciones de velocidad; traducir la lógica de diseño en instrucciones ejecutables requirieron énfasis en la programación; y el diseño de placas y circuitos reafirmó la importancia de la organización, la eficiencia y la reducción de ruido en sistemas electrónicos. Cada una de estas áreas concretaron la idea principal para dar vida al proyecto.

Mirando hacia el futuro, este artefacto de Persistencia de Visión ofrece un vasto potencial para futuras mejoras y expansiones. Una de las primeras evoluciones sería la incorporación de un mayor número de LEDs por centímetro en la tira, lo que incrementa drásticamente la resolución de las imágenes proyectadas, permitiendo detalles más finos y transiciones más suaves. A largo plazo, el objetivo sería que esta pantalla POV reproduzca imágenes mucho más complejas y, eventualmente, secuencias de vídeo completas. Esto implicaría desafíos significativos en el procesamiento de datos, la velocidad de transferencia al microcontrolador y la capacidad de actualización de los LEDs, pero sería un emocionante paso hacia la creación de pantallas volumétricas realmente inmersivas.

Bibliografia

- Datasheet led 5mm
  https://www.make-it.ca/5mm-led-specifications/
- Datasheet N555
  https://www.alldatasheet.com/datasheet-pdf/view/161282/TI/N555.html
- Datasheet A3144 sensor hall
  https://www.alldatasheet.es/datasheet-pdf/pdf/55092/ALLEGRO/A3144.html
- Referencia, ventilador holográfico
  https://www.youtube.com/watch?v=QdAjvYxWI1o
- persistencia de la visión
  https://www.masterclass.com/articles/persistence-of-vision-explained
- Datasheet mosfet IRFZ44N
  https://www.alldatasheet.es/datasheet-pdf/pdf/68619/IRF/IRFZ44N.html
- Controlador de velocidad con un temporizador
  https://how2electronics.com/light-dimmer-circuit-using-555-timer-ic-mosfet/
- Cálculos referidos al integrado ne555
  https://www.digikey.com.mx/es/resources/conversion-calculators/conversion-calculator-555-timer?srsltid=AfmBOoqYHh__HuPIIffhBO68JJTZjMUXg8kybd1UBrqrYNkASkKYAe0P
- Referencia para crear código
  https://maker.pro/arduino/projects/arduino-pov-display
- Tecnologia de proyeccion con el efecto POV
  https://luminafans.com/blogs/hologram-blog/how-hologram-fans-work?srsltid=AfmBOoqXhWd6Jtq-J58uShoXyO_e_QVjVTIP1r-YiZXrWiIlxHgn78Zp
- Cálculos para resistencias para los leds
  https://www.inventable.eu/paginas/LedResCalculatorSp/LedResCalculatorSp.html
- Modulo de carga inalámbrica
  https://ssdielect.com/cargadores-de-baterias-1/4632-xkt-412.html
- Motores para proyectos
  https://www.luisllamas.es/tipos-motores-rotativos-proyectos-arduino/
- Sobrecalentamientos en las placas impresas
  https://www.acdcecfan.com/es/pcb-heat-dissipation-techniques/

Anexo.1 pcb y esquemático

A. Controlador de velocidad

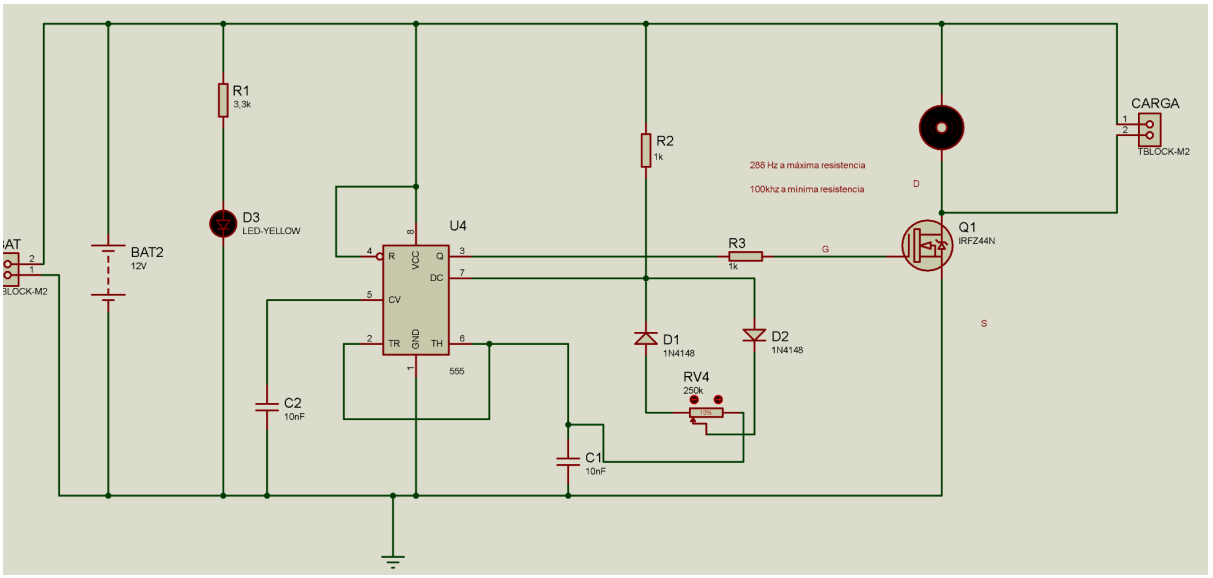


fig. 7.1 esquemático de control de velocidad

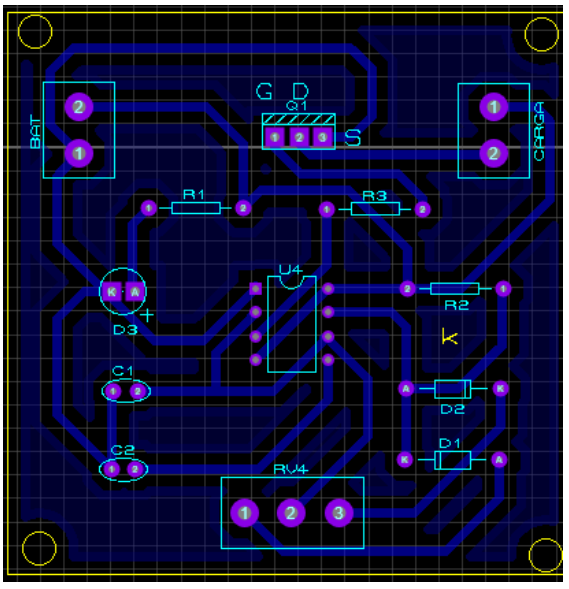


fig. 7.2 pcb del control de velocidad

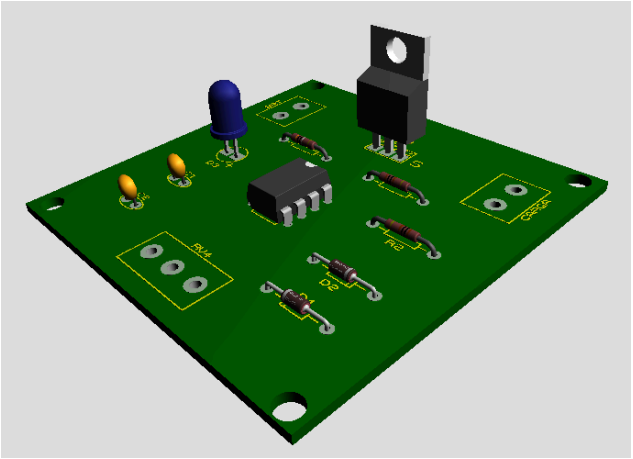


fig. .7.3 img. 3d del control de velocidad

B. Hélice de proyección

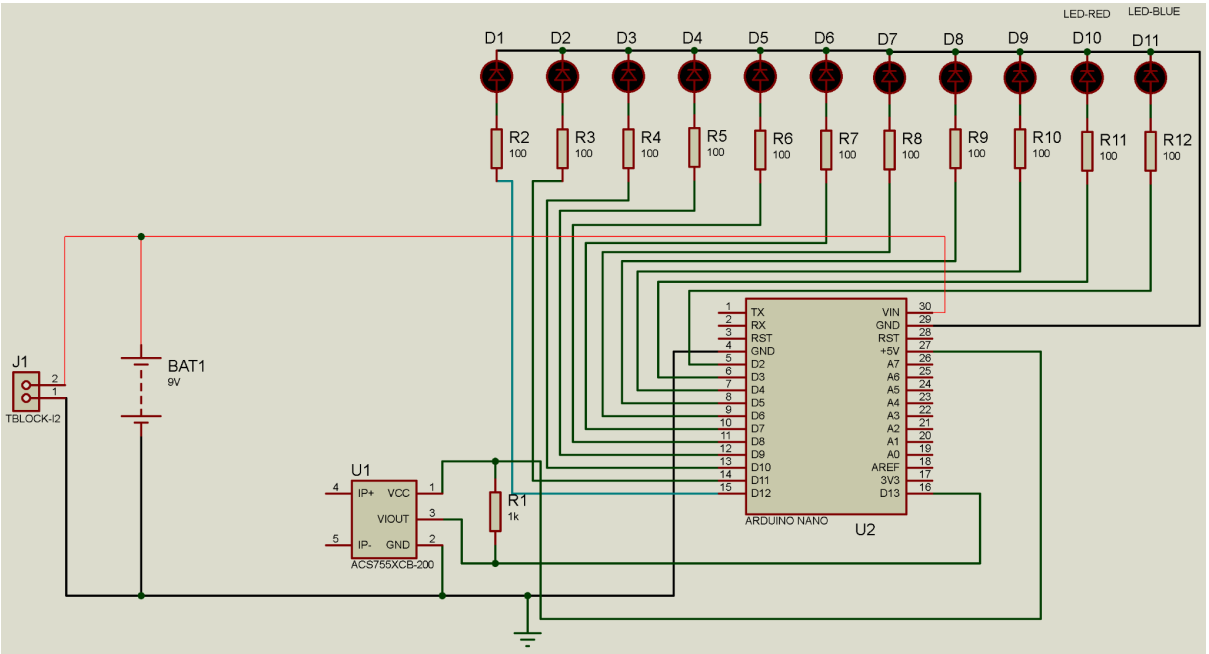


fig. 8.1 esquemático del módulo de proyección

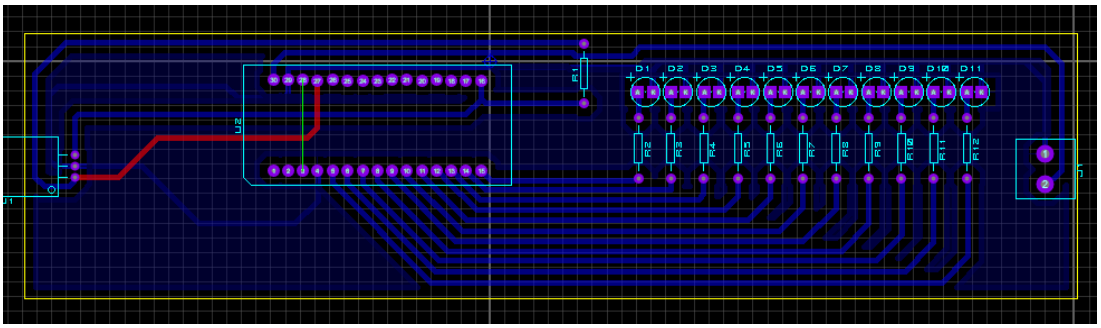


fig. 8.2 pcb del módulo de proyección

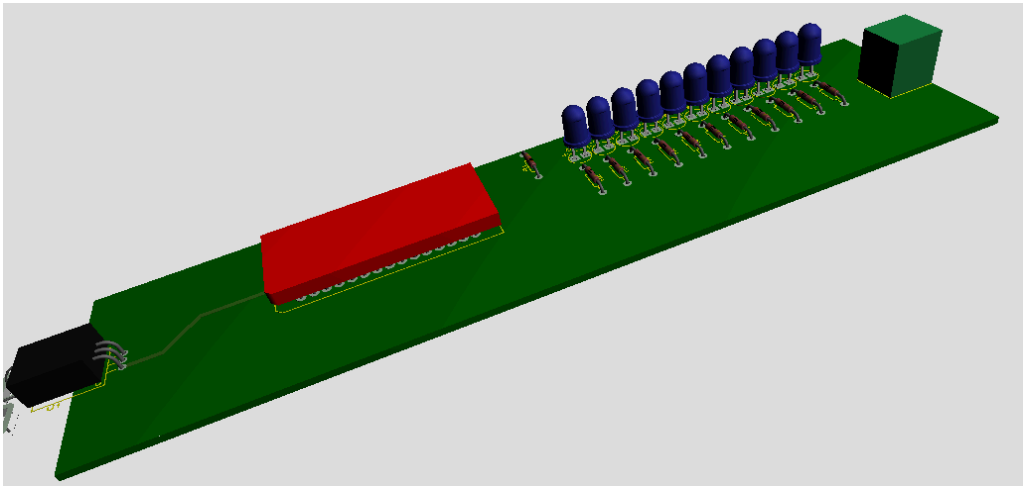


fig. 8.3 img. 3d del módulo de proyección

Anexo.2 codigo:

```
int text_ok=0;
int a[] = {126, 144, 144, 144, 126};
int b[] = {254, 146, 146, 146, 108};
int c[] = {254, 130, 130, 130, 130};
int d[] = {254, 130, 130, 130, 124};
int e[] = {254, 146, 146, 146, 146};
int f[] = {254, 144, 144, 144, 128};
int g[] = {124, 130, 138, 138, 76};
int h[] = {254, 16, 16, 16, 254};
int i[] = {130, 250, 130};
int j[] = {12, 2, 2, 2, 252};
int k[] = {254, 16, 40, 68, 130};
int l[] = {254, 2, 2, 2, 2};
int m[] = {254, 64, 32, 64, 254};
int n[] = {254, 32, 16, 8, 254};
int o[] = {124, 130, 130, 130, 124};
int p[] = {254, 136, 136, 136, 112};
int q[] = {124, 130, 138, 134, 126};
int r[] = {254, 144, 152, 148, 98};
int s[] = {100, 146, 146, 146, 76};
int t[] = {128, 128, 254, 128, 128};
int u[] = {252, 2, 2, 2, 252};
int v[] = {248, 4, 2, 4, 248};
```

```
int w[] = {254, 4, 8, 4, 254};
int x[] = {198, 40, 16, 40, 198};
int y[] = {224, 16, 14, 16, 224};
int z[] = {134, 138, 146, 162, 194};

int eos[] = {0, 3, 2,0};
int excl[] = {0, 250, 0};
int ques[] = {64, 128, 138, 144, 96};


float delayTime=1;



//POV clock cariables
unsigned long currentMillis, elapsed_loop_counter, previousMillis;
unsigned long counter_1, current_count;


// Variables de interrupción para contar la velocidad de rotación
// Creamos 4 variables para almacenar el valor anterior de la señal de
entrada (si es LOW o HIGH)
byte last_IN_state;               //Aquí almacenamos el estado anterior
en el pin digital 13
float one_rot_time=0;             //Aquí almacenamos el tiempo de
rotación completo.
float time_per_deg=0;             //Aquí almacenamos el tiempo que se
tarda en realizar una rotación de un grado.


void setup() {
  PCICR |= (1 << PCIE0);    //habilitar escaneo PCMSK0
  PCMSK0 |= (1 << PCINT5);  //Habilitar la interrupción del estado del
pin en el pin D13

  //Configuración de registro de pines de salida
  DDRD |= B11111100;      //2 a 7 como salida
  DDRB |= B00011111;      //8 a 12 como salida
  DDRB &= B11011111;      //13 enrada
  PORTD &= B00000011;     //2 a 7 LOW
  PORTB &= B00000000;     //8 a 13 LOW
}
```

```
void draw_a_line(int this_line){
int now_line;
now_line = this_line;
if (now_line>=128){PORTD |= B00100000;  now_line-=128;} else {PORTD &=
B11011111;}
if (now_line>=64) {PORTD |= B01000000;  now_line-=64;}  else {PORTD &=
B10111111;}
if (now_line>=32) {PORTD |= B10000000;  now_line-=32;}  else {PORTD &=
B01111111;}
if (now_line>=16) {PORTB |= B00000001;  now_line-=16;}  else {PORTB &=
B11111110;}
if (now_line>=8)  {PORTB |= B00000010;  now_line-=8;}   else {PORTB &=
B11111101;}
if (now_line>=4)  {PORTB |= B00000100;  now_line-=4;}   else {PORTB &=
B11111011;}
if (now_line>=2)  {PORTB |= B00001000;  now_line-=2;}   else {PORTB &=
B11110111;}
if (now_line>=1)  {PORTB |= B00010000;  now_line-=1;}   else {PORTB &=
B11101111;}
}
```

```
void displayChar(char cr, float line_delay){
if (cr == 'a'){for (int i = 0; i <5;
i++){draw_a_line(a[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'b'){for (int i = 0; i <5;
i++){draw_a_line(b[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'c'){for (int i = 0; i <5;
i++){draw_a_line(c[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'd'){for (int i = 0; i <5;
i++){draw_a_line(d[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'e'){for (int i = 0; i <5;
i++){draw_a_line(e[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
```

```
if (cr == 'f'){for (int i = 0; i <5;
i++){draw_a_line(f[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'g'){for (int i = 0; i <5;
i++){draw_a_line(g[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'h'){for (int i = 0; i <5;
i++){draw_a_line(h[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'i'){for (int itr = 0; itr <3;
itr++){draw_a_line(i[itr]);delayMicroseconds(line_delay);}draw_a_line(0
);}
if (cr == 'j'){for (int i = 0; i <5;
i++){draw_a_line(j[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'k'){for (int i = 0; i <5;
i++){draw_a_line(k[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'l'){for (int i = 0; i <5;
i++){draw_a_line(l[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'm'){for (int i = 0; i <5;
i++){draw_a_line(m[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'n'){for (int i = 0; i <5;
i++){draw_a_line(n[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'o'){for (int i = 0; i <5;
i++){draw_a_line(o[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'p'){for (int i = 0; i <5;
i++){draw_a_line(p[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'q'){for (int i = 0; i <5;
i++){draw_a_line(q[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'r'){for (int i = 0; i <5;
i++){draw_a_line(r[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 's'){for (int i = 0; i <5;
i++){draw_a_line(s[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 't'){for (int i = 0; i <5;
i++){draw_a_line(t[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'u'){for (int i = 0; i <5;
i++){draw_a_line(u[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'v'){for (int i = 0; i <5;
i++){draw_a_line(v[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'w'){for (int i = 0; i <5;
i++){draw_a_line(w[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'x'){for (int i = 0; i <5;
i++){draw_a_line(x[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'y'){for (int i = 0; i <5;
i++){draw_a_line(y[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
if (cr == 'z'){for (int i = 0; i <5;
i++){draw_a_line(z[i]);delayMicroseconds(line_delay);}draw_a_line(0);}
```

```
if (cr == '!'){for (int i = 0; i <3;
i++){draw_a_line(excl[i]);delayMicroseconds(line_delay);}draw_a_line(0)
;}
if (cr == '?'){for (int i = 0; i <5;
i++){draw_a_line(ques[i]);delayMicroseconds(line_delay);}draw_a_line(0)
;}
if (cr == '.'){for (int i = 0; i <4;
i++){draw_a_line(eos[i]);delayMicroseconds(line_delay);}draw_a_line(0);
}
delayMicroseconds(line_delay*2);
}
```

```
void displayString(char* s, float line_delay){
for (int i = 0; i<=strlen(s); i++){
displayChar(s[i],line_delay);
}
}
```

```
void loop() {

  currentMillis = micros();
  elapsed_loop_counter = currentMillis - previousMillis;
  delayTime = time_per_deg*3.5; //queremos 2 grados por cada línea de
las letras

  //Esto es para asegurarme de que comenzaré a imprimir a 216 grados
para que el texto quede centrado.
  if((elapsed_loop_counter >=  time_per_deg*(216)) &&
(elapsed_loop_counter <  time_per_deg*(217)) &&  text_ok)
  {
  displayString("electronica",delayTime); //AQUI CAMBIAMOS EL TEXTO DEL
LETRERO
  //delayMicroseconds(delayTime*10);
  text_ok=0;
  }
}
```

```
ISR(PCINT0_vect){
//Primero tomamos el valor de conteo actual en microsegundos usando la
función micros ()

  current_count = micros();
  //////////////////////////////////////Channel 1
  if(PINB & B00100000){                                  //Hacemos un AND
con el registro de estado del pin, ¿Verificamos si el pin 8 está ALTO
???
    if(last_IN_state == 0){                              //Si el último
estado fue 0, entonces tenemos un cambio de estado ...
      last_IN_state = 1;                                 //Almacene el
estado actual en el último estado para el siguiente bucle
      counter_1 = current_count;                          //Establezca el
counter_1 en el valor actual.
    }
  }
  else if(last_IN_state == 1){                           //Si el pin 8 es
LOW y el último estado fue HIGH, entonces tenemos un cambio de estado
    last_IN_state = 0;                                   //Almacene el
estado actual en el último estado para el siguiente bucle
    one_rot_time = current_count - counter_1;   //Nosotros hacemos la
diferencia de tiempo. Channel 1 es current_time - timer_1.
    time_per_deg = one_rot_time/360.0;
    previousMillis = micros();
    text_ok=1;
   }


}
```

Anexo.2.1 funcionamiento del código

¿Cómo se logran escribir cada letra?

Las letras se "dibujan" definiendo su forma con **arrays de números enteros**. Cada número en un array representa una "línea" vertical de la letra, y los bits de ese número controlan qué LEDs se encienden en esa línea.

Ejemplo:

```
int a [ ] = {126, 144, 144, 144, 126};
```

- **int a[] = {126, 144, 144, 144, 126};**: Esto define la letra 'A'. Es un array de 5 números enteros. Cada número es un patrón de bits que representa una columna vertical de LEDs. Imagina que el ventilador tiene 8 LEDs (o más, depende del hardware). Cada bit en el número binario controla si un LED está encendido (1) o apagado (0) en esa columna.
    - **126 en binario es 01111110**: Esto podría significar que el segundo LED desde arriba hasta el penúltimo están encendidos para la primera columna de la 'A'.
    - **144 en binario es 10010000**: Esto significaría que el primer y el cuarto LED están encendidos para la siguiente columna, y así sucesivamente.

La función draw_a_line(int this_line)

Esta función es el corazón del "dibujo". Toma un número entero (this_line) y lo descompone en sus bits para encender o apagar los LEDs correspondientes.

```
void draw_a_line(int this_line){
  int now_line;
  now_line = this_line;
  // Cada 'if' comprueba si un bit específico está encendido en
'now_line'
  // y luego enciende o apaga el LED correspondiente a través de los
registros del microcontrolador (PORTD, PORTB).
  // Por ejemplo, para el bit 7 (valor 128):
  if (now_line>=128){PORTD |= B00100000;  now_line-=128;} else {PORTD
&= B11011111;}
  // ... y así sucesivamente para los otros bits/LEDs
}
```

- **PORTD y PORTB**: Son los **registros de puertos de salida** del microcontrolador (probablemente un Arduino/ATmega328p). Al modificar estos registros directamente, se controlan los pines digitales (y por lo tanto, los LEDs conectados a ellos) de forma muy rápida y eficiente.
    - |= B00100000;: Establece un bit (pone un pin en HIGH) sin afectar los demás.
    - &= B11011111;: Limpia un bit (pone un pin en LOW) sin afectar los demás.

La función displayChar(char cr, float line_delay)

Esta función toma una letra (cr) y el retardo entre líneas (line_delay). Luego, dependiendo de la letra, itera sobre el array de esa letra y llama a draw_a_line() para cada "columna" de la letra.

```
void displayChar(char cr, float line_delay){
  if (cr == 'a'){
    for (int i = 0; i <5; i++){
      draw_a_line(a[i]); // Dibuja cada columna de la 'A'
      delayMicroseconds(line_delay); // Espera un tiempo para que la
columna sea visible
    }
    draw_a_line(0); // Apaga todos los LEDs al final de la letra
  }
  // ... similar para todas las demás letras
  delayMicroseconds(line_delay*2); // Espacio entre caracteres
}
```

- El bucle for recorre los números en el array de la letra.
- draw_a_line(a[i]); enciende los LEDs según el patrón de bits de esa columna.
- delayMicroseconds(line_delay); introduce un pequeño retraso. Este retraso es crucial porque, al girar el ventilador, cada columna se mostrará en una posición angular diferente. Un retraso más largo significa que la columna se mostrará más "ancha" angularmente.
- draw_a_line(0); al final de cada letra apaga todos los LEDs, creando un pequeño espacio entre las columnas de la misma letra y un espacio más grande entre letras.

displayString(char* s, float line_delay)

Esta función toma una cadena de caracteres (char* s) y simplemente llama a displayChar() para cada letra en la cadena.

```
void displayString(char* s, float line_delay){
  for (int i = 0; i<=strlen(s); i++){ // Recorre cada carácter de la
cadena
    displayChar(s[i],line_delay); // Dibuja el carácter
  }
}
```

¿Cómo se posiciona la palabra escrita según la velocidad de giro?

Aquí es donde entra en juego el **sensor Hall** y los cálculos de tiempo. El objetivo es que la palabra siempre aparezca en el mismo lugar y con el tamaño correcto, sin importar qué tan rápido gire el ventilador.

Variables clave:

- **one_rot_time**: Almacena el tiempo que tarda el ventilador en dar una **rotación completa (360 grados)** en microsegundos.
- **time_per_deg**: Calcula el tiempo que tarda el ventilador en girar **un grado** (one_rot_time / 360.0).
- **delayTime**: Este es el line_delay que se pasa a displayChar. Se calcula como time_per_deg * 3.5. Esto significa que cada columna de una letra se mostrará durante el tiempo que al ventilador le toma girar 3.5 grados. Esto controla el "ancho" de cada columna de la letra en el espacio holográfico.

La función loop() y el posicionamiento:

```
void loop() {
  currentMillis = micros(); // Tiempo actual en microsegundos
  elapsed_loop_counter = currentMillis - previousMillis; // Tiempo
transcurrido desde la última vez que el sensor Hall detectó una vuelta
completa

  delayTime = time_per_deg*3.5; // Calcula el tiempo de retardo para
cada línea/columna

  // Esto es para asegurarme de que comenzaré a imprimir a 216 grados
para que el texto quede centrado.
  if((elapsed_loop_counter >=  time_per_deg*(216)) &&
(elapsed_loop_counter <  time_per_deg*(217)) &&  text_ok)
  {
    displayString("electronica",delayTime); // Dibuja la palabra
    text_ok=0; // Evita que la palabra se dibuje continuamente en la
misma vuelta
  }
}
```

- **elapsed_loop_counter = currentMillis - previousMillis;**: Este contador mide cuánto tiempo ha pasado desde que el sensor Hall detectó el punto de inicio de una rotación (previousMillis se actualiza en la interrupción del sensor Hall).
- **if((elapsed_loop_counter >= time_per_deg*(216)) && (elapsed_loop_counter < time_per_deg*(217)) && text_ok)**: Esta es la condición clave para el posicionamiento.

    - **time_per_deg*(216)**: Calcula el tiempo que tardaría el ventilador en llegar a 216 grados desde el punto de inicio (donde se activó el sensor Hall).
    - El if dice: "Si el tiempo transcurrido (elapsed_loop_counter) es **aproximadamente cuando el ventilador ha girado 216 grados** Y text_ok es verdadero (lo que significa que una nueva vuelta ha comenzado y aún no hemos dibujado la palabra para esta vuelta), entonces **dibuja la palabra**."
    - El valor 216 (o un valor similar) se elige para que la palabra se muestre en una posición visualmente adecuada cuando el ventilador gira. Probablemente, 0 grados es donde se activa el sensor Hall, y 216 grados es el ángulo donde quieren que la palabra empiece a aparecer para que esté "centrada" o bien posicionada desde la perspectiva del observador.
- **text_ok=0;**: Una vez que la palabra se ha dibujado para esta rotación, text_ok se pone a 0. Esto evita que la palabra se dibuje múltiples veces durante la misma rotación, lo que causaría que la imagen se vea borrosa o repetida. Solo se dibuja una vez por cada rotación completa.

¿Cómo utiliza la señal de un sensor Hall para poder escribir la palabra en cada vuelta?

El sensor Hall es fundamental porque proporciona un **punto de referencia** para cada rotación. Cuando el imán del ventilador pasa por el sensor Hall, este genera un pulso (cambia su estado de LOW a HIGH o viceversa). Este pulso es detectado por el microcontrolador a través de una **interrupción**.

setup() para las interrupciones del sensor Hall:

```
void setup() {
  PCICR |= (1 << PCIE0);    // Habilitar escaneo PCMSK0 (Pin Change
Interrupt Control Register)
  PCMSK0 |= (1 << PCINT5);  // Habilitar la interrupción del estado del
pin en el pin D13 (PCINT5 corresponde a D13)

  // ... (configuración de pines de salida)
}
```

- **Interrupciones de Cambio de Pin (PCINT)**: El microcontrolador puede ser configurado para que, en lugar de estar constantemente revisando el estado de un pin (lo que sería ineficiente), "salte" a una función específica (una Interruption Service Routine o ISR) cada vez que el estado de un pin cambie.
- **PCICR |= (1 << PCIE0);**: Habilita el grupo de interrupciones de cambio de pin PCIE0, que incluye los pines del puerto B (donde está D13).
- **PCMSK0 |= (1 << PCINT5);**: Habilita la interrupción para el pin específico D13 (PCINT5).

ISR(PCINT0_vect) - La rutina de servicio de interrupción:

```
ISR(PCINT0_vect){ // Esta función se ejecuta automáticamente cuando el estado del pin D13 cambia
  current_count = micros(); // Obtiene el tiempo actual

  if(PINB & B00100000){ // Si el pin D13 (PCINT5, bit 5 del PORTB) está HIGH
    if(last_IN_state == 0){ // Y el estado anterior era LOW (detecta un flanco de subida)
      last_IN_state = 1;    // Actualiza el estado
      counter_1 = current_count; // Guarda el tiempo cuando el pin se puso HIGH
    }
  }
  else if(last_IN_state == 1){ // Si el pin D13 está LOW Y el estado anterior era HIGH (detecta un flanco de bajada)
    last_IN_state = 0;      // Actualiza el estado
    one_rot_time = current_count - counter_1; // Calcula el tiempo total de una rotación
    time_per_deg = one_rot_time/360.0; // Calcula el tiempo por grado
    previousMillis = micros(); // Marca este momento como el inicio de una nueva rotación para el bucle principal
    text_ok=1; // Indica que se puede dibujar la palabra en la próxima oportunidad
  }
}
```

- **Detección de flanco**: La ISR detecta cuándo el sensor Hall pasa de LOW a HIGH (o de HIGH a LOW, dependiendo de cómo esté configurado el sensor y el imán). En este caso, parece que mide el tiempo entre un flanco de subida y un flanco de bajada del sensor Hall (o entre dos flancos de subida si el imán es suficientemente grande).
- **one_rot_time = current_count - counter_1;**: Esta es la línea clave. Cuando el sensor Hall detecta el "fin" de su pulso (cambio de HIGH a LOW), calcula la duración de ese pulso, que se utiliza como la duración de una rotación completa. **Esto asume que el pulso del sensor Hall representa una rotación completa**, o que es un punto fijo dentro de la rotación para calcular el tiempo total de una vuelta.
- **time_per_deg = one_rot_time/360.0;**: Divide el tiempo total de la rotación por 360 para saber cuánto tiempo tarda el ventilador en girar un solo grado. ¡Esta es la información crítica para la sincronización!.
- **previousMillis = micros();**: Almacena el tiempo actual como el punto de inicio para la próxima rotación en el loop().

- **text_ok=1;**: Una vez que se calcula un nuevo tiempo de rotación, se establece text_ok en 1. Esto le dice a la función loop() que en la próxima rotación del ventilador, la palabra debe ser dibujada.

En resumen, el sensor Hall le dice al microcontrolador **cuándo comienza una nueva rotación** y, al medir el tiempo entre esas activaciones, **calcula la velocidad de giro**. Con esta velocidad, el código sabe exactamente cuánto tiempo esperar entre el encendido de cada LED (columna de la letra) para que, a medida que el ventilador gira, los puntos de luz se unan para formar las letras en el aire, siempre en la misma posición angular.

Anexo.3 Imágenes del proceso de elaboración

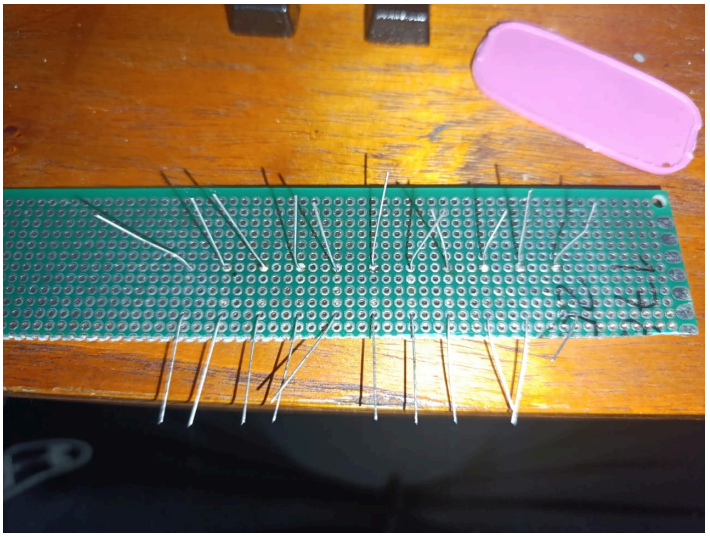


fig. 9.1 fotografia de la helice

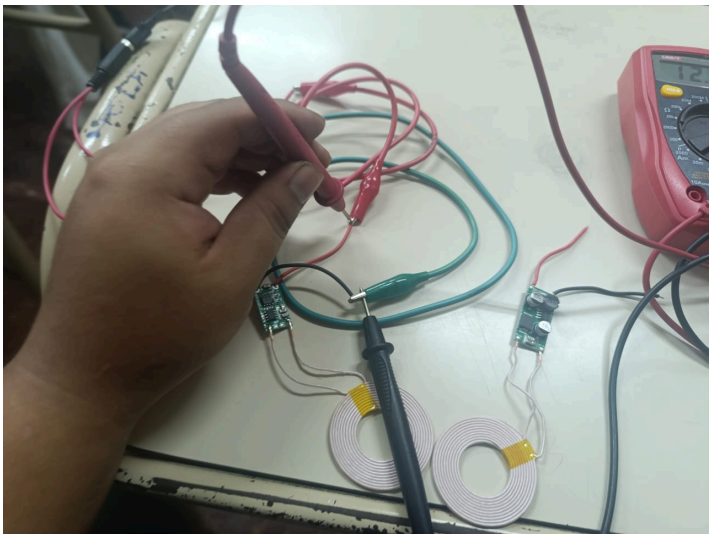


fig. 9.2 fotografía de prueba de tensión

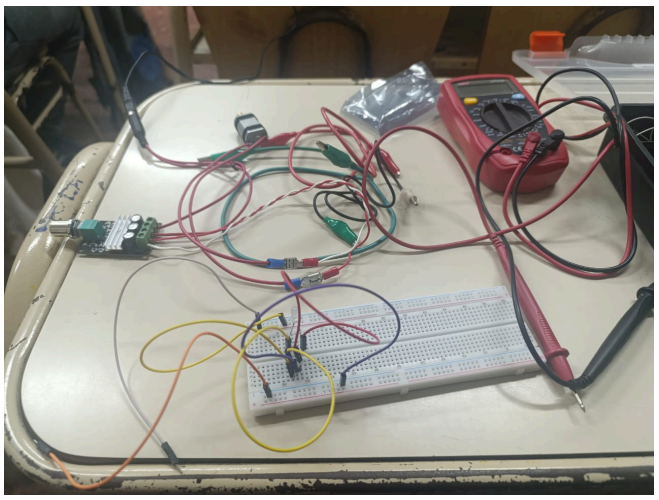


fig. 9.3 fotografía de voltaje

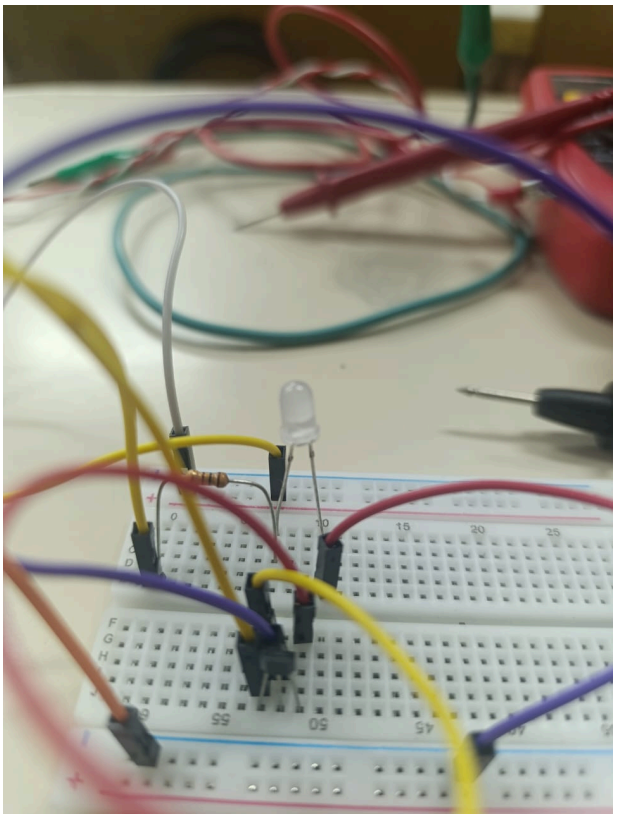


fig. 9.4 fotografia del sensor hall

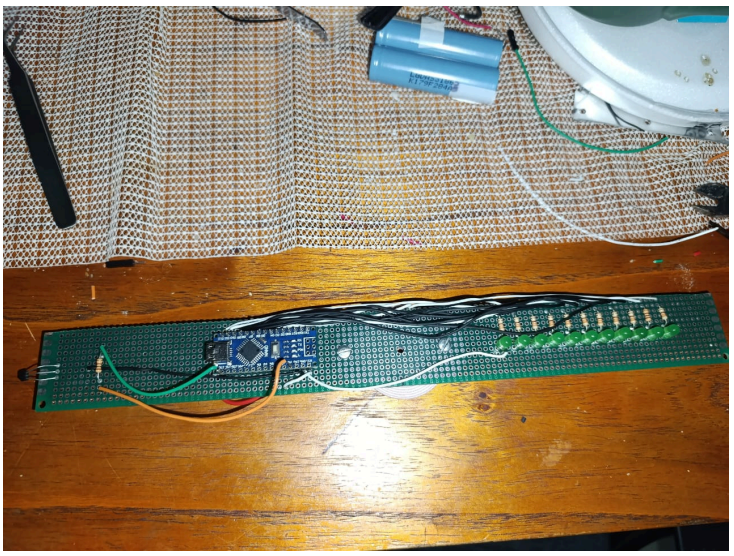


fig. 9.5 fotografia helice terminada

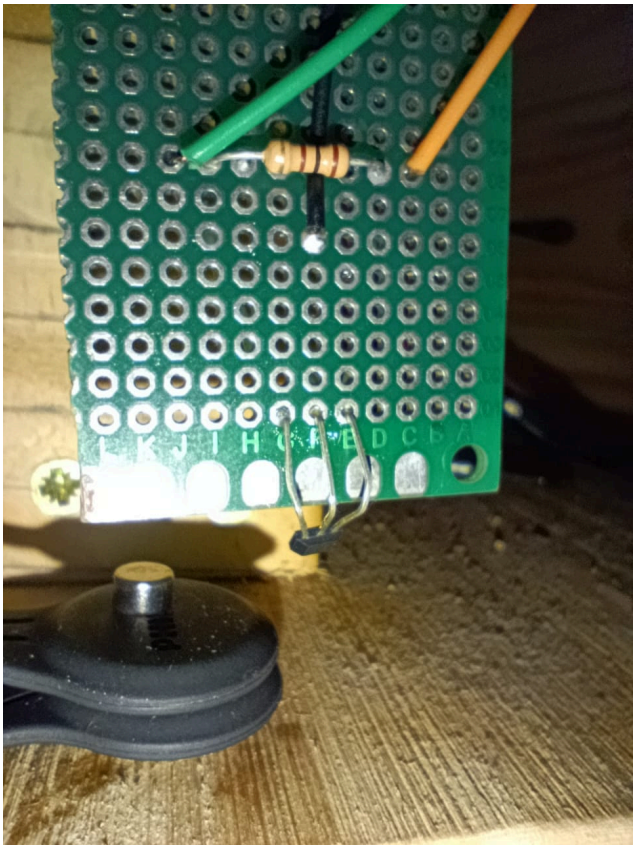


fig. 9.6 fotografía posicionamiento sensor hall

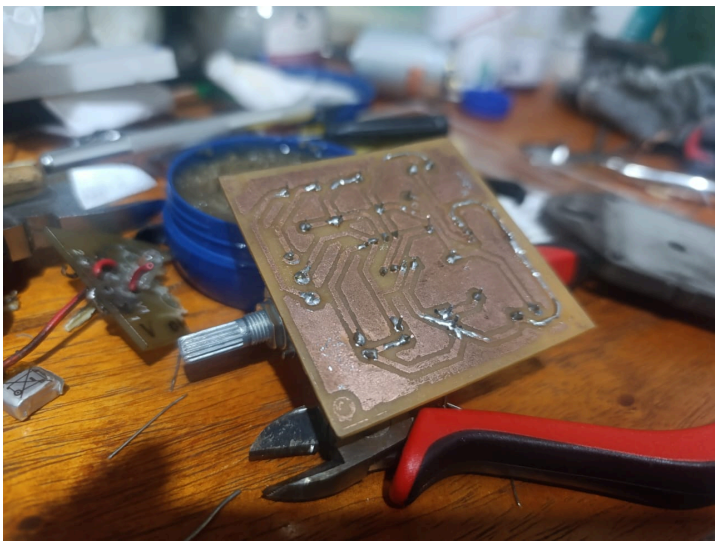


fig. 9.7 fotografía de placa del control de velocidad

fig. 9.8 fotografía con error en la proyección

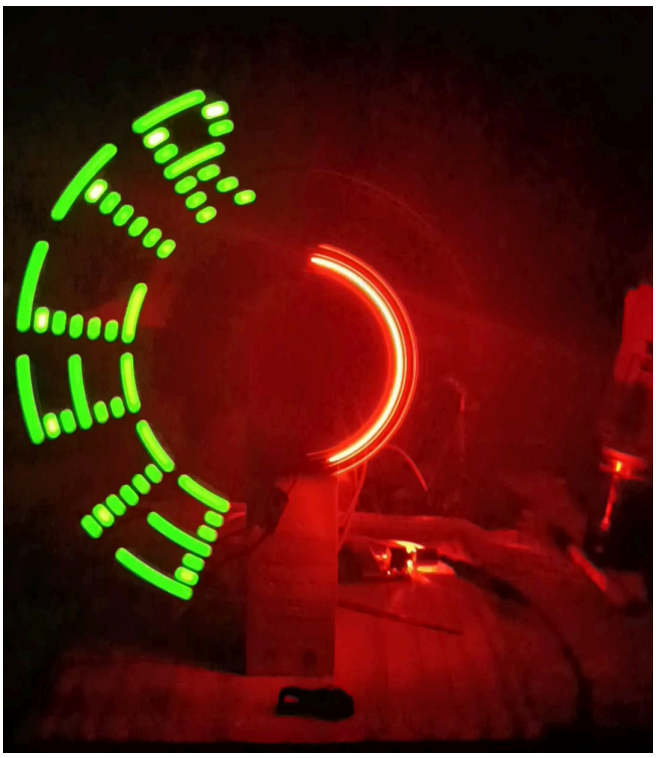


fig. 9.9 fotografía de la proyección

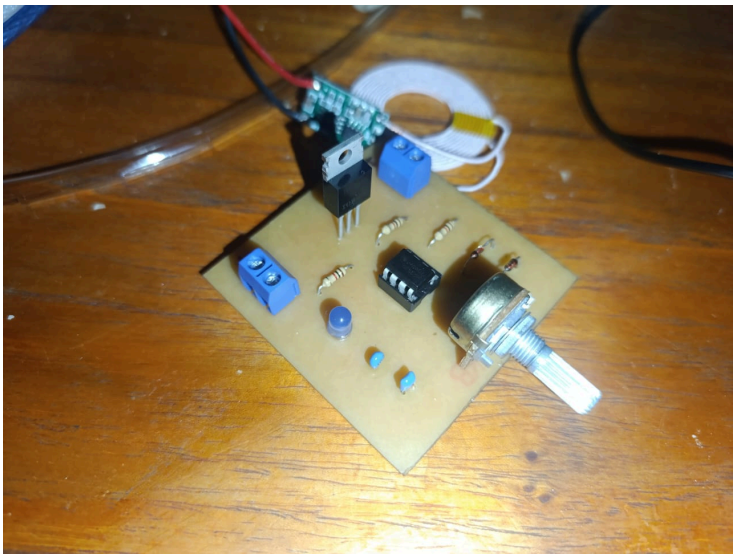


fig. 9.10 fotografía del regulador de velocidad

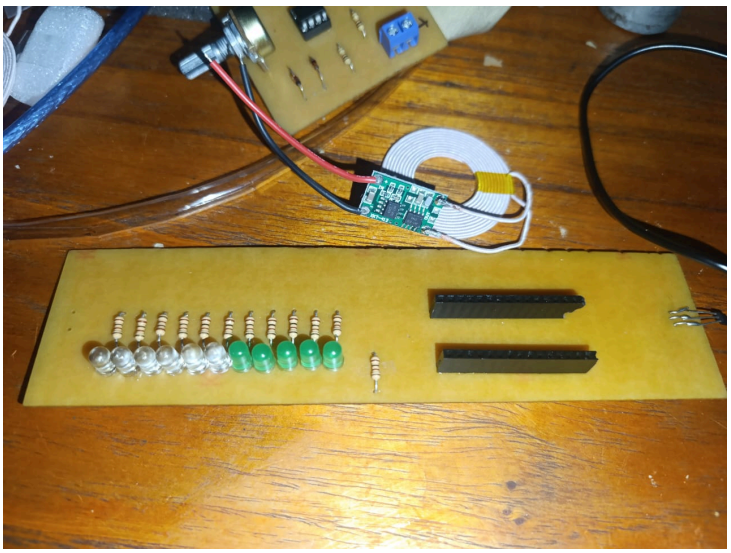
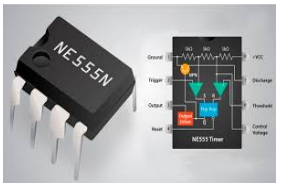
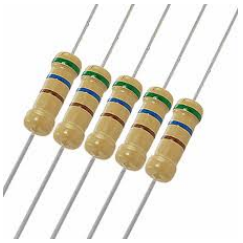
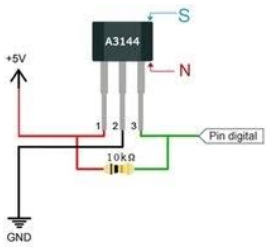
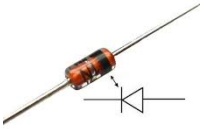
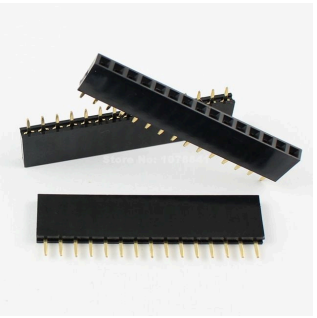


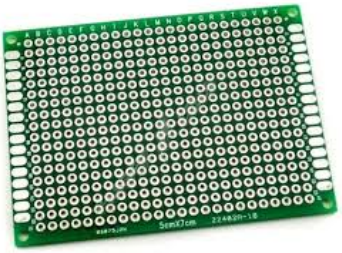
fig. 9.11 fotografía de prototipo de hélice

Anexo.4 precio de componentes

Componentes	Precio (dolar)	Imagen
Integrado NE555	$2.11	
Led verde 5mm	10 un.$3	

Resistencias	50 un. $3.24	
Sensor hall A3144	5 un. $3.41	
Capacitores	10 un. $3.18	
Diodos	2 un. $2.7	
Potenciometro	2 un. $4.5	

Mosfet IRFZ44N	$2.3	
Pines hembra	2 un. $2.76	
Bornera de dos pines	2 un. $4.62	
Zocalo ne555	$0.81	

Led azul 5mm	10 un.$3	
Placa perforada	$5.59	
Motor dc 12v	$5.68	
Emisor/ Receptor de carga inalambrica	$13.78	

Microcontrolador arduino nano	$2.84	
Borneras	$2.11	
Cables 0.5mm	$6.49	
Tabla de madera	$4.22	

Fuente Switching Cargador 12v 1a Amp	$7.30	
Ficha hembra jack 5.5	$1.38