ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Diseño y fabricación de un escáner 3D sin contacto basado en las plataformas Arduino y Matlab

(Design and fabrication of an Arduino and Matlab-based contactless 3D scanner)

Para acceder al Título de *Graduado en Ingeniería de Tecnologías de Telecomunicación*

Autor: Alberto Villa Fernández

Octubre -2020

Contenido

1 Planteamiento

1.1 Introducción

El objetivo de este proyecto es diseñar e implementar un sistema capaz de realizar un escaneo tridimensional de un objeto de pequeñas dimensiones. Para ello, el sistema tendrá que almacenar la imagen tridimensional del objeto a través de una matriz de puntos en coordenadas cartesianas.

Se dispondrá de distintos formatos de escaneo dependiendo de los requisitos del usuario respetando un compromiso entre el tiempo y la calidad. Por un lado, el escaneo con mayor calidad dispondrá de un mayor número de puntos junto a una velocidad de procesado lenta, mientras que el escaneo de menor calidad actuará unos tiempos de ejecución mucho menores.

Para realizar las funciones indicadas, el sistema se compondrá por varios dispositivos separados en dos bloques: el bloque de control de giro y el bloque de procesado de datos. Ambos bloques se interconectan mediante una comunicación de tipo puerto serie.

En primer lugar, el bloque de control de giro se compone de una estructura cilíndrica, un motor paso a paso, un láser y una placa Arduino. La estructura cilíndrica se utilizará como base donde posicionar el objeto que se desea escanear. Esta base dispondrá de una plataforma central diseñada para rotar 360º sobre una guía de rodamientos que facilitan el giro de la plataforma mientras reducen el rozamiento. El motor paso a paso, conectado físicamente a la plataforma móvil,

transfiere a esta el movimiento rotatorio, mientras que su control se realiza a través de la placa Arduino Uno.

El láser, ubicado a una distancia fija de la estructura, y en paralelo a la cámara de adquisición de imágenes, emitirá un haz de luz vertical sobre el objeto. El análisis de las características de este haz, fotografiado por la cámara, será la tarea del bloque de procesado de datos. Por último, la placa Arduino controlará la corriente de entrada del motor, logrando una alta precisión en cada rotación, además de encargarse del procesado de las instrucciones de puerto serie y la alimentación de los otros elementos del bloque.

En segundo lugar, el bloque de procesado de datos se compone de un ordenador con el software de Matlab que ejecute el script de Matlab y una Webcam operativa conectada al mismo.

Antes de continuar con la resolución del proyecto, se ha de introducir el concepto de escaneado 3D. Se definirá, de aquí en adelante, al escaneado 3D como el proceso de analizar un objeto del mundo real para obtener datos sobre su forma y aspecto. El objetivo de este proceso es la completa digitalización de la forma del objeto para su uso en variedad de aplicaciones como son la captura de movimiento, la ingeniería inversa, el diseño industrial de prótesis o el control de calidad.

Dentro de las tecnologías de escaneo 3D existen dos aproximaciones dependiendo de la necesidad o no de contacto físico, tal y como se resume en la Tabla 1.

En primer lugar, los escáneres de contacto se basan en apoyar un elemento palpador sobre la superficie del objeto. Mediante el uso de sensores de presión internos, se obtiene una medida de la posición espacial del objeto. Esta opción aporta una gran precisión a la medida, típicamente alrededor de los 0,01 mm. Sin embargo, su mayor problema es el requerimiento de contacto físico con el objeto, imposibilitando escanear objetos delicados, como pueden ser circuitos impresos u obras históricas.

Por otra parte, los escáneres sin contacto no necesitan de contacto físico con el objeto para funcionar y se basan en distintas tecnologías a distancia como pueden ser los ultrasonidos o pulsos de luz. Dada la naturaleza de los objetos a escanear, y a la sencillez de su implementación (su bajo coste y la amplia disponibilidad de material), se ha optado por la realización de un escáner 3D sin contacto basado en la aproximación tecnológica de luz estructurada. Esta aproximación se basa en la emisión de una fuente de luz coherente (láser) sobre el objeto a escanear, mientras que una cámara detecta la luz reflejada por el objeto.

La implementación más sencilla de escaneo 3D por luz estructurada utiliza la proyección de un punto de luz circular sobre el objeto para recoger un único punto de iluminación en la lente de la cámara. Mediante la posición relativa de dicho punto respecto a la fuente de luz y a la lente de la cámara, es posible determinar la ubicación en el espacio tridimensional del punto, obteniendo su localización en coordenadas cartesianas. Este método llevaría al sistema a analizar cada punto por separado, realizando un barrido vertical y otro horizontal de todo el objeto, provocando un tiempo de análisis demasiado grande como para considerarse viable.

En este trabajo de Fin de Grado se ha optado por proyectar un haz de luz lineal sobre el objeto, que permita iluminar una línea desde su base hasta su extremo superior. En este caso únicamente será necesario rotar el objeto respecto al eje vertical para completar el escaneo. Así pues, la resolución del sistema estará limitada por la anchura del haz y la calidad óptica de la cámara.

Alternativamente existen varias estrategias para reconstruir una imagen tridimensional mediante un único análisis, con una calidad mucho menor, con el objetivo de analizar objetos en movimiento. En este caso, se generan patrones de rayas mediante dos fuentes de luz lo que genera interferencias equidistantes. El análisis de los patrones resultantes lleva a una aproximación rápida, pero de baja precisión (20 mm) de la forma del objeto [1].

	De contacto	Luz Estructurada	
		Rendija	Patrones
Precisión	0.01 mm	1 mm	20 mm
Velocidad	Lenta	Rápida	Instantánea

Tabla 1. Características Tecnologías.

En la Fig. 1 pueden apreciarse cada uno de los distintos elementos que componen el sistema de escaneo 3D. Siendo estos identificados como: ordenador con software Matlab (A), LED conectado a Arduino (B), webcam (C), placa Arduino (D), microcontrolador (E) y estructura rotatoria con motor integrado (F).

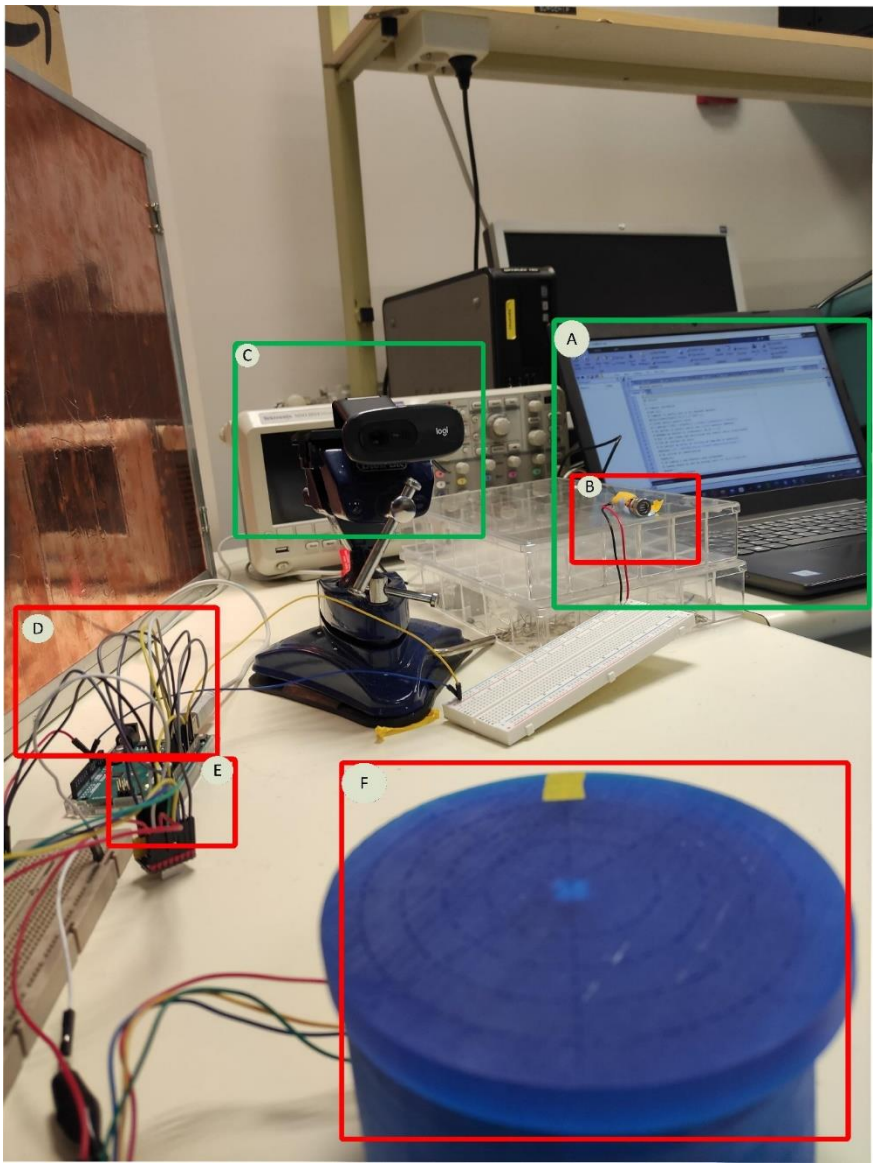


Fig. 1. Fotografía del sistema de escáner 3D completo.

2 Sistema

En esta parte de la memoria se explicarán en profundidad todos los elementos que conforman el proyecto. En primer lugar, se describen los componentes físicos como son la webcam, la placa Arduino o el motor paso a paso. Seguidamente, se realiza una descripción teórica de cada paso del algoritmo de procesado de imagen que se ejecutará, desde la calibración hasta la generación de coordenadas tridimensionales. Por último, se incorpora una descripción del código a ejecutar en el script de Matlab y en la ejecución de la placa Arduino.

2.1 Componentes

Los componentes utilizados para implementar el sistema pueden agruparse en aquellos destinados al bloque de control de giro, y los pertenecientes al bloque de procesado de datos. Una agrupación de todos estos elementos puede observarse en la Fig. 2.

En primer lugar, desde el bloque de procesado de datos se dispondrá de una cámara webcam con la que capturar imágenes del objeto. En segundo lugar, se desarrollará una estructura monitorizada por una placa Arduino que se conecta a un motor paso a paso. El objetivo de esta estructura se centra en controlar el giro de rotación de la plataforma mediante la ejecución de instrucciones de Matlab vía el puerto serie de Arduino.

Para lograr una rotación suficientemente precisa en cada paso del motor se utiliza un microcontrolador que permitirá multiplicar por 16 el número de pasos necesarios para completar una rotación. Este procesador estará controlado mediante pulsos eléctricos de las salidas digitales de la placa Arduino.

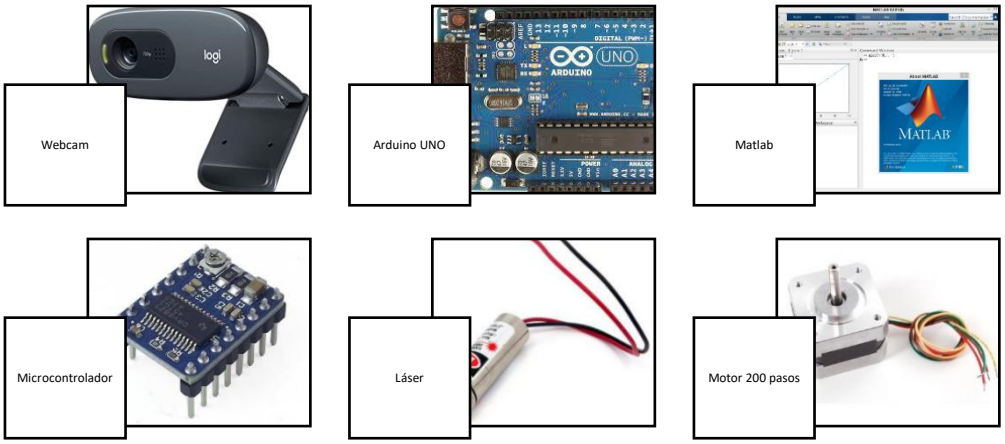


Fig. 2. Partes del sistema.

El bloque de control de giro se compone de la placa Arduino, el motor y el láser. El láser genera un haz lineal que apunta sobre la plataforma rotatoria motorizada, siendo ambos controlados mediante la placa Arduino.

El bloque de procesado de datos se compone de un PC con software Matlab® y una cámara web. La cámara web toma instantáneas de la luz reflejada por el objeto, y el computador con el software Matlab® se encarga de procesar las imágenes y estimar la posición en el espacio de la sección vertical iluminada del objeto. Esta posición se almacenará en una matriz en forma de coordenadas cartesianas.

2.1.1 Webcam

Para obtener una imagen digital del objeto es necesario disponer de una cámara controlable desde el software Matlab®, utilizado para el procesado de datos.

En este sentido el uso de cámaras webcam Plug and Play está contemplado por la herramienta Matlab® webcam, la cual detecta de forma automática como entrada de imagen dicha cámara como webcam y almacena los datos en una matriz 1280 × 720 × 3 píxeles. Donde la resolución de la captura es de 1280 pixeles horizontales por 720 pixeles verticales

Se escoge el modelo Logitech C270 HD para esta aplicación debido a que la resolución que nos proporciona de 1280 × 720 es más que suficiente para apreciar la línea de puntos que forma el contorno de un objeto de pequeñas dimensiones.

La velocidad de captura de la cámara es de 30 frames por segundo. Esta velocidad de captura es lo suficientemente alta como para no ralentizar el proceso de escaneado, por lo que este elemento no se identifica como un elemento crítico en el momento de optimizar el algoritmo de escaneo.

2.1.2 Arduino Uno

Para el manejo y control de la plataforma de giro se utilizará una placa Arduino Uno la cual se comunica mediante un puerto serie con Matlab®. La placa Arduino permitirá controlar el microcontrolador, dando la opción de escoger opciones como la velocidad de rotación

(seleccionando el número de pasos hasta un total de 1600 pasos mediante $microstepping$), el sentido del giro y el control de flujo de corriente sobre el motor evitando el consumo innecesario de potencia cuando este no es utilizado.

Según la página web de Arduino, se define la placa Arduino como [2]:

> ***"Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.".***

La placa se compone de un pequeño procesador, una memoria y varios pines de entrada/salida que puedes controlar mediante la programación IDE de Arduino. Estos pines controlaran el microcontrolador del motor paso a paso y la corriente en el diodo LED.

La placa se debe programar cada vez que se conecta a corriente debido a que el programa compilado se encuentra en una memoria $flash$. Esta compilación y carga del programa se realizará desde el programa Arduino con el puerto serie.

2.1.3 Matlab

Para el cálculo y procesamiento de los datos se utilizará el programa informático Matlab® en su versión R2019b, el objetivo final es programar una interfaz de usuario GUI y la implementación de un algoritmo recursivo que modele la nube de puntos obteniendo un modelado 3D de la imagen.

Matlab® es un sistema de cómputo que ofrece un IDE con un lenguaje de programación propio. Se utilizarán diferentes toolbox durante el proceso siendo estas:

- **Computer Vision Toolbox v9.1:**

  Se trata de una toolbox muy extensa de la que se utilizará únicamente el script de calibración de cámaras para obtener los parámetros intrínsecos de cualquier cámara a utilizar.

  La toolbox proporciona algoritmos, funciones y $apps$ para el diseño y la realización de pruebas de sistemas de procesamiento de vídeo, visión artificial y visión 3D. Es posible llevar a cabo la detección y seguimiento de objetos, así como la detección y extracción de características [3].

- **MATLAB® Support Package for USB Webcams v19.2.2:**

  Esta toolbox complementa el sistema de Matlab® haciendo accesible el uso de USB Webcams [4].

- **Image Acquisition Toolbox v6.1:**

  Mediante esta toolbox se obtendrá la matriz de cada captura realizada con la webcam convirtiendo está en datos manejables para el algoritmo.

Esta toolbox proporciona funciones y bloques para conectar cámaras a Matlab® y Simulink®. Incluye una app de Matlab® que permite detectar y configurar de forma interactiva las propiedades del hardware. A continuación, puede generar código Matlab® equivalente para todas las sesiones próximas. Permite conectar con cámaras de todo tipo, permitiendo modos de adquisición en segundo plano, activación de hardware y sincronización mediante varios dispositivos [5].

2.1.4 Motor paso a paso

Se escoge un motor de continua de 200 pasos convencional con pasos de 1,8º de rotación, mediante el microcontrolador se realizará un proceso de microstepping logrando rotaciones con precisión de hasta 0.05625º por paso, permitiendo realizar una vuelta completa con 6400 pasos.

El modelo escogido es el SparkFun Electronics ROB-09238®. Se trata de un motor hibrido bipolar con cuatro terminales y dos bobinas muy utilizado en este tipo de proyectos. Se ha escogido por su sencillez dentro del espectro de los motores bipolares y por su reducido precio en comparación a otros motores unipolares que podrían resultar más sencillos.

El motor se alimenta con 12V de tensión y una corriente de 330 mA, consumiendo una potencia de 4 W de forma constante, por lo que se deberá regular el consumo en estado de espera para evitar la disipación de calor, al encontrarse el motor ubicado dentro de una plataforma cilíndrica de plástico creada por impresión 3D se deberá hacer hincapié en evitar altas temperaturas para no deteriorar la infraestructura.

2.1.5 Microcontrolador

Se utilizará un microcontrolador DRV8825 de Texas Instruments para aplicar la indexación microstepping con el objetivo de aumentar al máximo la resolución de nuestro objeto, obteniendo un numero de pasos de 200, 400, 800, 1600, 3200 o 6400, permitiendo escaneados más rápidos de resolución angular 1,8º o escaneos lentos con mucha más resolución angular 0.05625º.

Más adelante en el documento se explicará de forma exhaustiva el funcionamiento del microcontrolador desde las señales de control hasta las señales eléctricas del motor. Datos obtenidos de su datasheet correspondiente [6].

2.1.6 Láser

Se usa un láser de 5mW de Adafruit Industries LLC con forma de haz para marcar la línea de puntos en cada iteración del proceso, sus características son las siguientes [7]:

	Mínimo	Típico	Máximo
Potencia	2,5 mW	3,0 mW	5,0 mW
Corriente	10 mA	20 mA	25 mA
Voltaje	2,3 V	4.5 V	8 V
Longitud de onda		650 nm	
Temperatura	-20ºC		+25ºC

Tabla 2. Características Láser.

Se usa este láser particular de helio-neón como los utilizados en punteros láser comerciales debido a la seguridad de este, la aplicación que se busca es una mera referencia para el sistema.

El punto más crítico del láser se encuentra en la rendija óptica del láser que regulará la precisión del sistema a la hora de detectar puntos reflejados.

2.2 Algoritmos de procesado de datos

2.2.1 Calibración

Para la aplicación del algoritmo se han de conocer varios datos de la cámara como es la distancia focal y las coordenadas del punto central de la cámara en píxeles, estos datos se obtienen mediante un proceso de calibración con la herramienta Computer Vision de Matlab®, que nos facilita conocer estos datos con mayor precisión que la dada por el fabricante.

La herramienta obtiene los parámetros de una lente para corregir la distorsión de la cámara y poder obtener el tamaño de un objeto en unidades del mundo real.

Se introducen una serie de capturas desde distintos ángulos y alturas de un patrón de cuadros previamente conocido, como el que se aprecia en la Fig. 3. El programa determina un patrón de resultados y se extrapola con la posición real del objeto, a menor diferencia en los resultados se obtienen unos resultados más precisos.

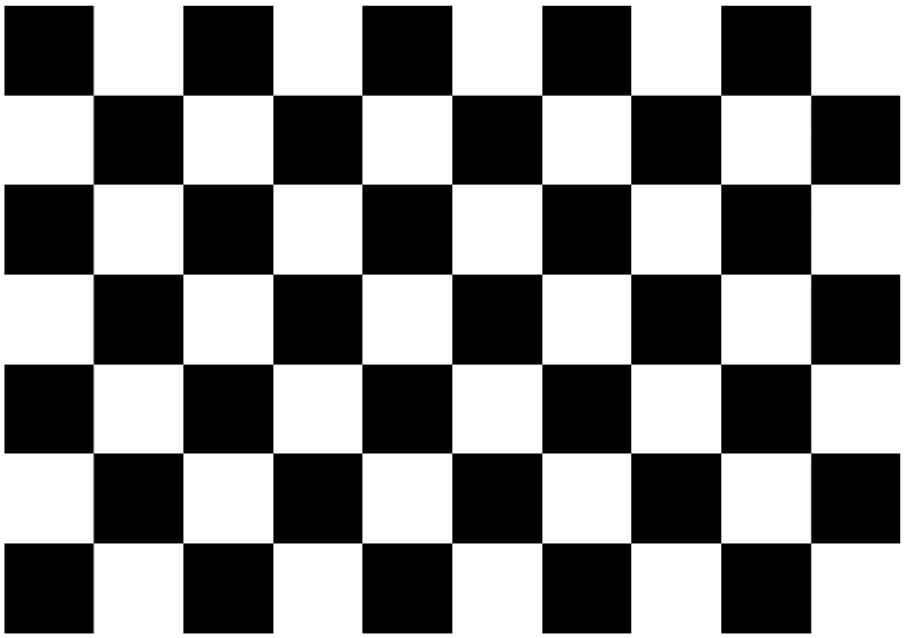


Fig. 3. Patrón de cuadros utilizado.

Con el objetivo de maximizar la precisión de la calibración se realiza una criba de resultados con las imágenes de menor error en la diferencia de posición, resultando en un error máximo próximo a los 0,5 pixeles.

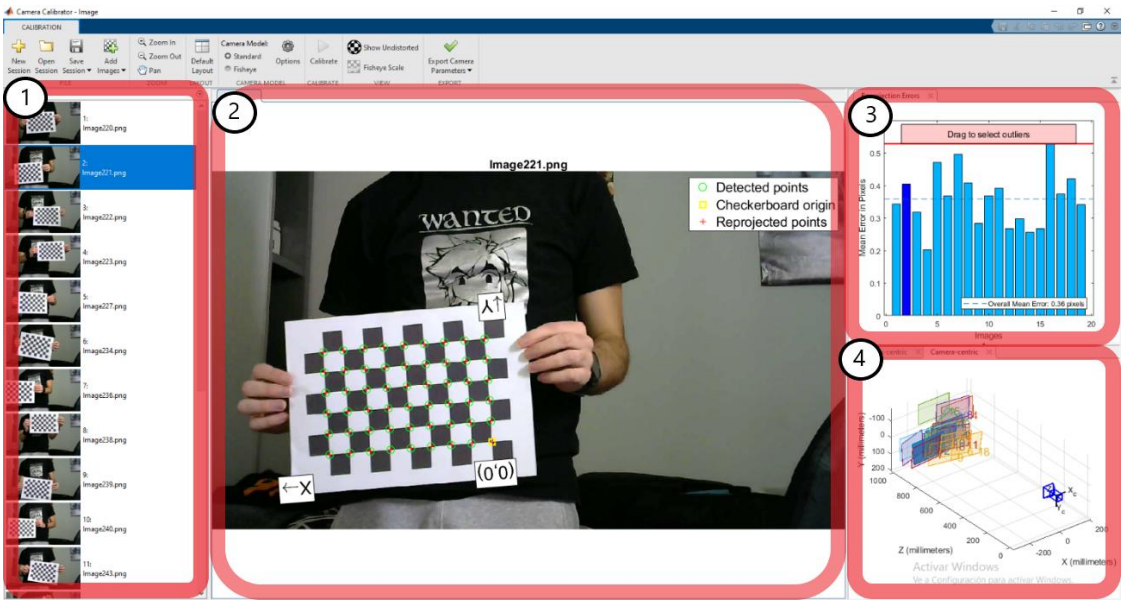


Fig. 4. Herramienta Computer Vision.

Como se aprecia en la Fig. 4, la herramienta se compone de una primera ventana (1) ubicada a la izquierda de la pantalla la cuál almacena las capturas realizadas. En la ventana central (2) muestra individualmente una de las capturas, indicándose la referencia inicial tomada, los puntos analizados y el sentido de los ejes X e Y de referencia. En la ventana superior derecha (3) se representa el error que genera cada captura en relación con los valores finales obtenidos, gracias a esta ventana es posible seleccionar qué imágenes deben ser analizadas para optimizar los valores resultantes que proporcionan la máxima precisión posible. Por último, la ventana inferior derecha (4) muestra la referencia tridimensional de la relación de imágenes capturadas considerando que la cámara se encuentra en un punto fijo.

La herramienta proporciona como salida los valores intrínsecos y extrínsecos de la cámara a calibrar permitiendo aplicar el tratamiento trigonométrico del algoritmo desarrollado para la nube de puntos. Este conjunto de datos, similar al incluido en la Tabla 3, permite contar con la información necesaria para aplicar el algoritmo de procesado de datos para obtener la posición espacial de un objeto escaneado.

Parámetro	Valor obtenido
Distancia Focal Eje X	1314,2 pixeles
Distancia Focal Eje Y	1318,1 pixeles
Punto Central Eje X	598,7328 pixeles
Punto Central Eje Y	301,5177 pixeles

Tabla 3 Valores Intrinsecos Cámara Logitech C270 HD

2.2.2 Cámara *pinhole*

Con el objetivo de desarrollar un algoritmo que permita obtener distancias reales a partir de una imagen captada por una cámara se va a analizar el funcionamiento de una cámara *pinhole*, también llamada estenopeica. Una cámara estenopeica es una cámara sin lente que dispone de una

pequeña abertura. La luz procedente del objeto observado pasa a la cámara oscura a través de la abertura y se proyecta una imagen invertida en el lado opuesto de la caja.

El desarrollo de la cámara oscura data del siglo XVII, cuando Athanasius Kircher diseña en un grabado similar a la Fig. 5 el funcionamiento de una cámara portátil de grandes dimensiones. Dicho grabado representa un cubo exterior del tamaño de una habitación, con un orificio y una lente en cada una de sus paredes, y en su interior otro prisma construido por pantallas de papel tensado y transparente, sobre el que la persona situada en su interior podía dibujar las escenas proyectadas desde el exterior [8].

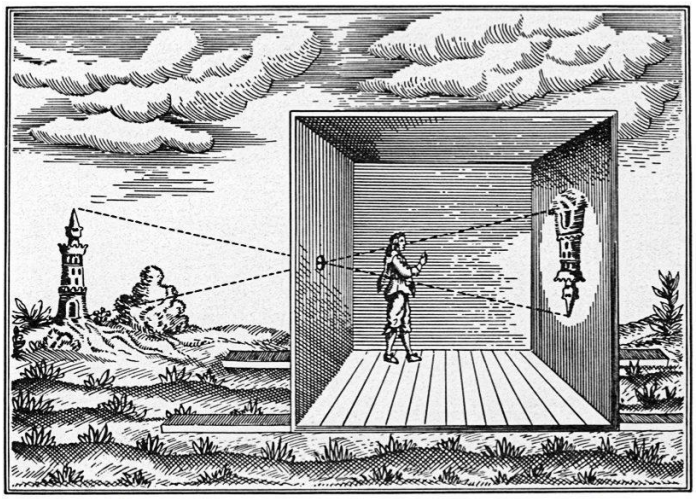


Fig. 5. Ilustración que representa la cámara *pinhole* del siglo XVII.

De forma simplificada, la Fig. 6 muestra el esquema del funcionamiento de una cámara *pinhole* con lente. El análisis de este esquema permitiría conocer las dimensiones del objeto (Δx_0) a partir de su representación en el sensor de la cámara (Δx_1).

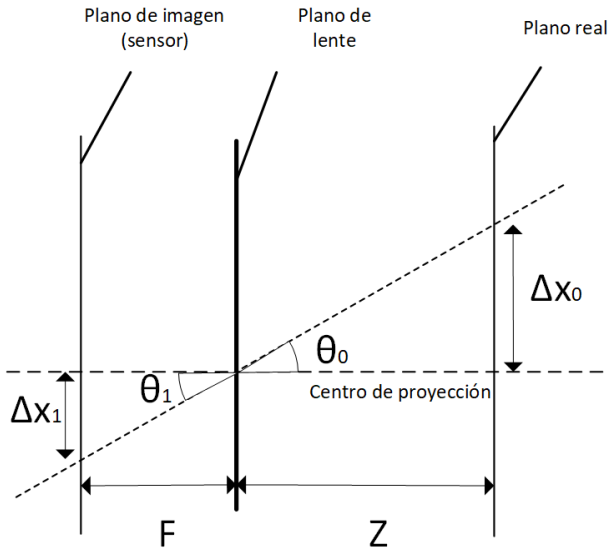


Fig. 6. Esquema simplificado del funcionamiento de una cámara *pinhole*.

El esquema mostrado incluye tres planos, correspondientes a la ubicación del objeto (plano "real"), la ubicación de la lente de la cámara (plano de "lente"), y la ubicación del sensor de la

cámara (plano de "imagen"). La separación entre dichos planos se indica en el esquema con las variables Z (distancia lente-objeto) y F (distancia sensor-lente ó distancia focal).

Adicionalmente, la distancia de objeto a fotografiar respecto al plano de proyección se indica como Δx_0 y representa unidades de longitud. Por otro lado, la distancia de su imagen en el sensor respecto al mismo plano (centro) de proyección se indica con la variable Δx_1 la cual tiene unidades de pixeles.

Finalmente, el rato de luz proveniente del objeto incide en el plano de lente con un ángulo θ_0 respecto al centro de proyección; mientras que ese mismo rayo incide en el plano del sensor formando un ángulo θ_1 en el plano de lente respecto al centro de proyección.

Por simple trigonometría conocemos que los dos ángulos que se oponen formados al cortarse dos rectas son iguales, por lo que puede afirmarse que:

$$\theta_0 = \theta_1 \tag{1}$$

Por lo tanto, se pueden relacionar las tangentes de ambos ángulos entre sí, llegando a una relación entre las distancias de los planos y el tamaño del objeto en el plano real y el plano de imagen.

$$\frac{\Delta x_0}{Z} = \tan(\theta_0) \tag{2}$$

$$\frac{\Delta x_1}{F} = \tan(\theta_1) \tag{3}$$

$$\frac{\Delta x_0}{Z} = \frac{\Delta x_1}{F} \tag{4}$$

La distancia real de cada punto del objeto (Z) dependerá de la distancia focal (F), su posición en el eje X relativa (Δx_1) y su posición en el eje X real (Δx_0). Esta relación viene dada por la ecuación de la cámara $pinhole$:

$$F(pixeles) = \frac{\Delta x_1(pixels) \cdot Z(mm)}{\Delta x_0(mm)} \tag{5}$$

Dicha expresión se utilizará en el procesado de datos para relacionar las distancias en planos "real" y planos de "imagen", así como para realizar la calibración inicial de la cámara.

2.2.3 Indexación de la imagen

Con el objetivo de relativizar las posiciones de la matriz de datos de la imagen capturada respecto a su punto central (0,0), es necesario realizar cambios en la indexación de la matriz respecto al formato utilizado por Matlab®. En este caso, la imagen adquirida se almacena en una matriz de 720 filas y 1280 columnas, situando al píxel de la esquina superior izquierda de la imagen; tal y como se puede apreciar en la Fig. 7.

Debe tenerse presente que Matlab® es un programa optimizado para realizar operaciones con matrices, y representa su sistema de indexación predeterminado. Sin embargo, dicha indexación no es intuitiva para realizar el procesado de datos, ya que se requiere que el pixel central de la imagen corresponda con la posición matricial (0,0).

Este cambio es necesario ya que Matlab® de forma predeterminada implementa un punto en el origen (1,1) en la esquina superior izquierda de la imagen, debido al funcionamiento de la indexación predeterminada de Matlab®.

Se demostrará de forma empírica que la cantidad de pixeles en la nueva indexación sea el correcto, para ello se ha de tener en cuenta la posición inicial y final de cada columna de pixeles modificada con la nueva asignación. Si bien las posiciones genéricas de Matlab inician en la posición $x_0 = 1$ y finalizan en la posición $x_{max} = 1280$, la nueva asignación deberá corregir la posición inicial $x = 1$. Para ello se desplazan todas las filas una posición hacia la derecha. Así mismo, se realiza la misma operación con las columnas que inician en la posición $y_0 = 1$ y terminan en la posición $y_{max} = 720$.

$$x_{inicializado} = x_{original} - 1$$

$$y_{inicializado} = y_{original} - 1$$

A continuación, se deberá centrar el origen en el punto intermedio de la imagen, desplazando filas y columnas en función de la posición central, se reasignan los índices de la posición central de la imagen como:

$$x_{central} = \frac{x_{max} - 1}{2}$$

$$y_{imagen} = \frac{y_{max} - 1}{2}$$

Se obtienen los valores del centro de la imagen como $x_{central} = 639$ e $y_{central} = 359$. Se ha de hacer hincapié en que estas coordenadas no corresponden exactamente el centro de la imagen, el centro de la imagen se encuentra en el punto intermedio de los 4 pixeles centrales (0,0),(0,1),(1,0) y (1,1). Esto se debe a la paridad del número de pixeles que compone la imagen y la inexistencia de un píxel central como tal.

Para poder centrar la imagen se reindexarán las columnas y filas inicializadas hasta que la posición inicializada (0,0) se encuentre en la posición original situada en el punto (639,359).

$$x_{imagen} = x_{inicializado} - 639 = x_{original} - 640$$

$$y_{imagen} = y_{inicializado} - 359 = y_{original} - 360$$

Con el objetivo de ilustrar la indexación realizada se muestra a continuación la ilustración 3 la cual refleja el posicionamiento de los pixeles centrales y los pixeles que se encuentran en las posiciones limite que se encuentran en las esquinas de la imagen.

X=-478 Y=-359					X=480 Y=-359
		X=0 Y=0	X=1 Y=0		
		X=0 Y=1	X=1 Y=1		
X=-478 Y=360					X=480 Y=360

Fig. 7. Esquema indexado de la matriz imagen.

2.2.4 Trigonometría

Se deberá de estudiar la relación que forman las líneas entre el láser y la línea de visión de la cámara, para ello se ha de enfatizar varias relaciones trigonométricas con el objetivo de clarificar las operaciones que serán necesarias para la resolución del sistema.

2.2.5 Descripción del sistema de coordenadas

Se plantea el sistema de coordenadas representando las posiciones de láser y cámara enfrentadas al objeto y la pared de referencia en una posición posterior en el eje Z. Un esquema de dichas posiciones se representa en la Fig. 8.

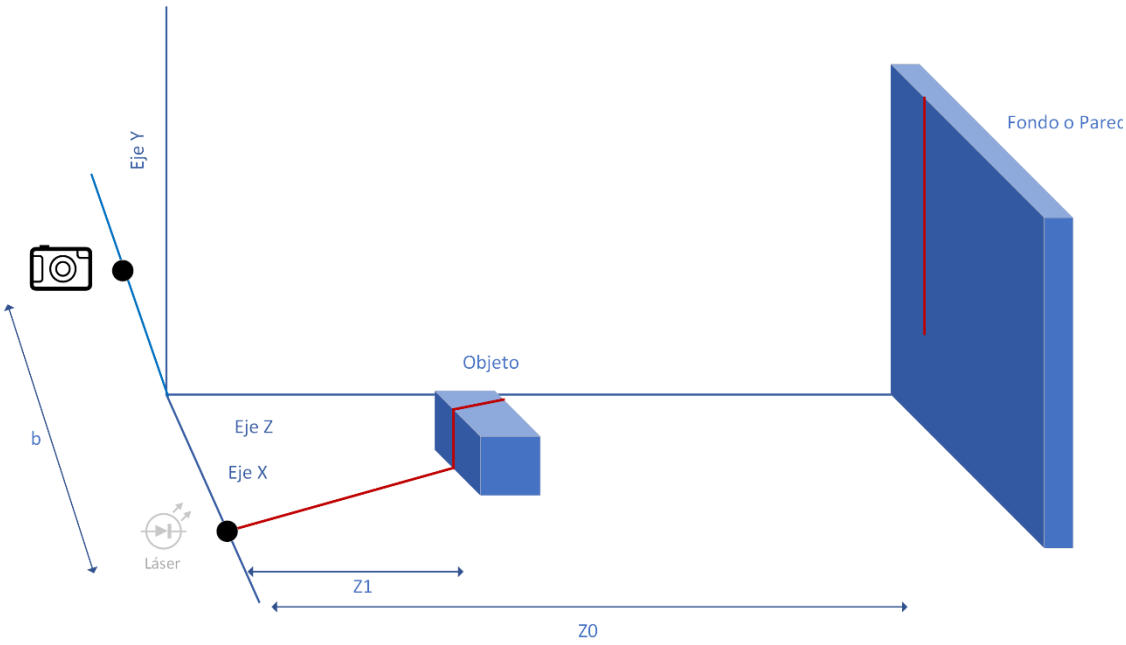


Fig. 8. Representación del eje de coordenadas y posicionamiento de la cámara, el láser, el objeto y la pared de referencia.

El objetivo de aplicar la trigonometría es calcular el valor de la distancia Z1 del objeto respecto al origen de coordenadas, para cualquiera de los puntos de su altura en el eje y. Los únicos parámetros conocidos son la distancia focal (F), la distancia del eje de coordenadas al fondo ($Z0$) y la distancia en el eje X entre la cámara y el láser (b). La distancia focal es obtenida mediante la calibración previa, este parámetro varía según la cámara utilizada y la resolución configurada.

2.2.6 Vista del plano ZX

Se ha de analizar la imagen mediante vista cenital, desde el plano XZ, visto en la Fig. 9. Estudiando en profundidad el plano se observan los siguientes ángulos. La línea del láser representada por la recta a_0, forma un ángulo C_0 con el plano de la cámara y crea una intersección con el plano del objeto y el plano pared, formando los puntos a analizar por la cámara. El segmento de la recta a_0 que inicia en el plano de la cámara y finaliza en la intersección de esta con el plano del objeto es denominado a_1.

Las próximas líneas que se analizan son las rectas c_0y c_1, son las rectas que forma el objetivo de la cámara con los puntos formados en las intersecciones entre el láser y los planos de objeto y de pared. Estas rectas se originan en el objetivo de la cámara con dos ángulos distintos A_0 y A_1. Para completar el triángulo se denominan los ángulos B_0y B_1, que son los ángulos formados por la intersección entre las rectas c_0 y a_0, y las rectas c_1y a_1.

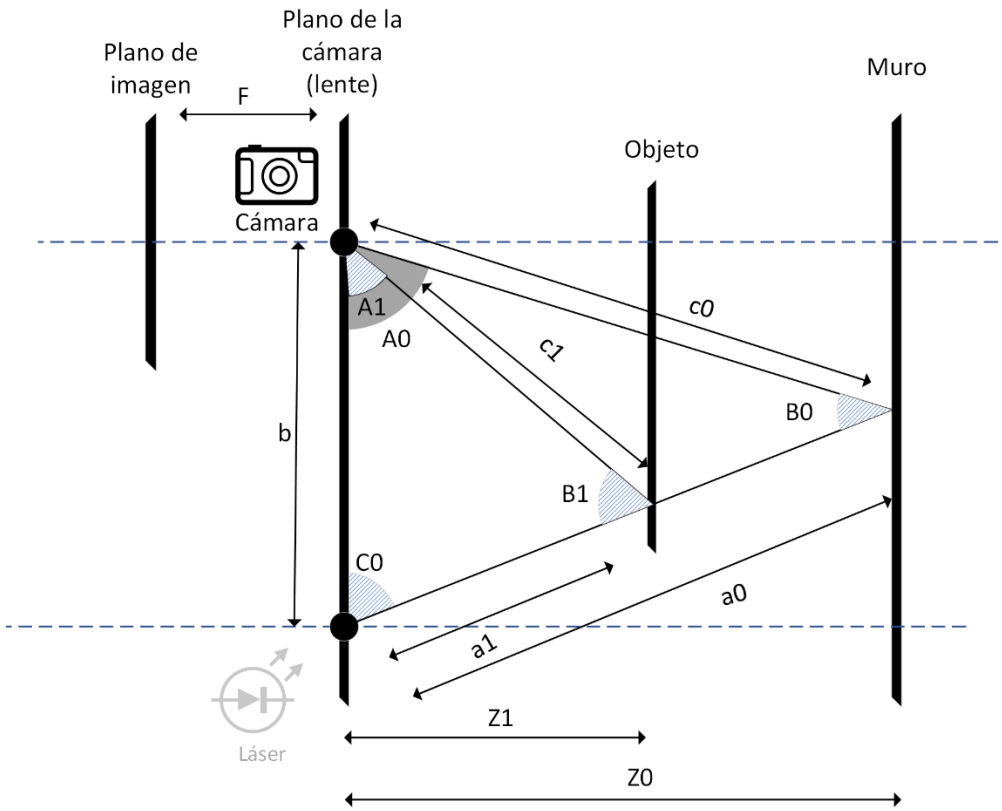


Fig. 9. Esquema del sistema en 2D, vista occipital desde el plano XZ.

2.2.7 Cálculo de los ángulos de la lente A0 y A1

Estudiando el plano de imagen, como el representado en la Fig. 10, puede calcularse el ángulo A_0 y A_1, se tendrá en cuenta la distancia focal para calcular dichos ángulos a partir de las relaciones de cámara $pinhole$ ya mencionadas anteriormente. Para ello se estudia la distancia en

el plano de imagen x_0' que simula la incidencia de un punto de luz laser desde la lente de la cámara y la distancia focal ya mencionada con anterioridad.

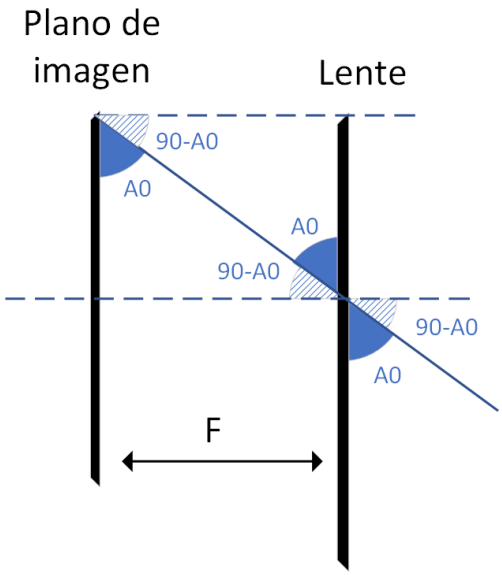


Fig. 10. Esquema de formación de la imagen en el sensor.

Por simple trigonometría obtenemos la relación entre los ángulos A_0 y A_1 con las distintas posiciones de los puntos de láser proyectados dentro del plano de imagen con el cual opera el algoritmo.

$$x_0' = h \cdot \cos(A_0) \tag{6}$$

$$F = h \cdot \sin(A_0) \tag{7}$$

Relacionando ambas funciones:

$$\tan(A_0) = \frac{\sin(A_0)}{\cos(A_0)} = \frac{\frac{F}{h}}{\frac{x_0'}{h}} = \frac{F}{x_0'} \tag{8}$$

Así, despejando la variable A_0:

$$A_0 = \tan^{-1}\left(\frac{F}{x_0'}\right) \tag{9}$$

Por tanto, los ángulos A_0 y A_1 pueden calcularse mediante la siguiente ecuación.

$$A_0 = \tan^{-1}\left(\frac{F}{x_0'}\right) = 90 - \tan^{-1}\left(\frac{x_0'}{F}\right) \tag{10}$$

$$A_1 = \tan^{-1}\left(\frac{F}{x_1'}\right) = 90 - \tan^{-1}\left(\frac{x_1'}{F}\right) \tag{11}$$

Donde los valores x_0', x_1' y F son valores conocidos, siendo coordenadas de la imagen en pixeles digitalizadas en la matriz imagen y el valor F obtenido en pixeles mediante la previa calibración.

2.2.8 Cálculo de relaciones con la pared de referencia

Se analizará en profundidad el sistema de triángulos formado por el trazado del láser y la posición del objeto con respecto a la posición estática de la cámara. Dicho sistema es representado en la Fig. 11. El objetivo final es caracterizar completamente el triángulo pared-laser-cámara.

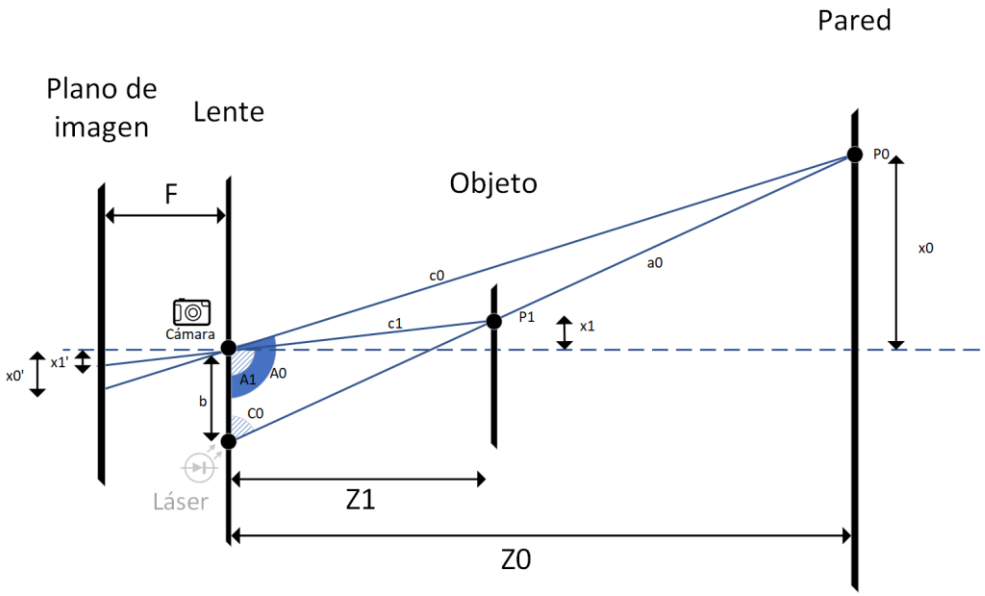


Fig. 11. Análisis trigonométrico de las relaciones entre los elementos del sistema.

En primera instancia, se va a detallar el triángulo que forma el láser cuando se ve reflejado en la posición estática P_0, en la distancia z_0, una distancia conocida de antemano siendo esta la distancia Cámara-Pared. Se caracterizará por completo el triángulo obteniendo los valores $x_0(mm)$, $c_0(mm)$, $a_0(mm)$, $B_0(°)$ y C(º). Dicho triángulo se ve interpretado por la Fig. 12.

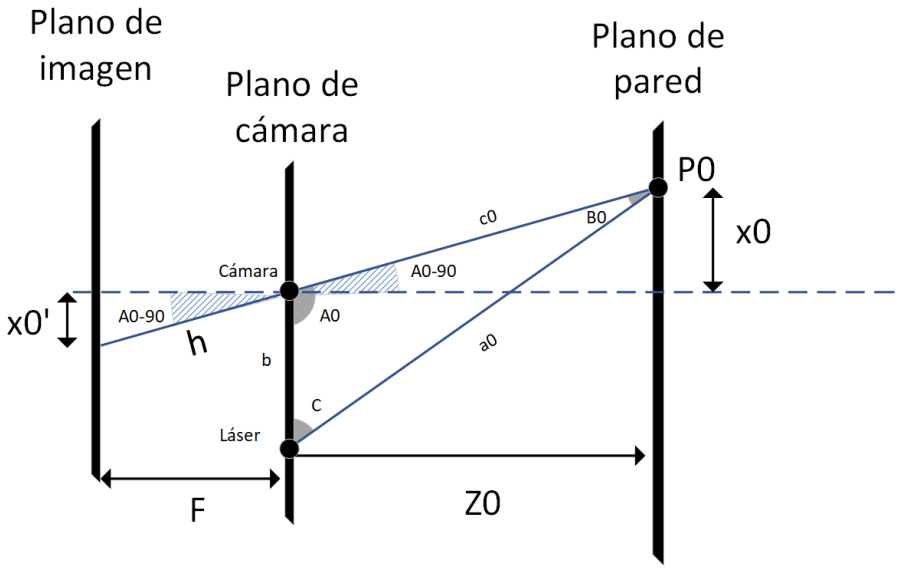


Fig. 12. Análisis de relaciones trigonométricas entre la tripla láser-cámara-pared.

Se analiza el sistema con un sistema de ecuaciones, obteniendo el valor de A_0 en esta configuración:

$$x_0' = h \cdot sin(A_0 - 90) \tag{12}$$

$$F = h \cdot cos(A_0 - 90) \tag{13}$$

Relacionando ambas expresiones (12) y (13) se llega a la ecuación (14):

$$\tan(A_0 - 90) = \frac{x_0'}{F} \tag{14}$$

Despejando el valor del ángulo A_0 $mediante\ la\ función\ arcotangente$:

$$A_0 = 90 + \tan^{-1}\left(\frac{x_0'}{F}\right) \tag{15}$$

Se analizará al final de este apartado la validez de la última expresión para la posibilidad de que el ángulo A_0 sea un ángulo agudo, menor de 90º, se ha generalizado la opción de ángulo obtuso, mayor de 90º, pero se demostrará que ambas situaciones se resuelven mediante el mismo procedimiento debido a las características de la función arcotangente.

Se aplican las propiedades de la cámara $pinhole$ para obtener el valor de x_0 a partir del valor ${x_0}'$ que obtenemos directamente de la imagen. Para este cálculo son necesarias la distancia focal en pixeles y la distancia z_0 real (distancia cámara-pared) en milímetros.

$$x_0 = \frac{{x_0}'(pixels) \cdot z_0(mm)}{F(pixels)} \tag{16}$$

Se continúa caracterizando al punto P_0, se conocen los lados del triángulo equilátero que conforman el eje horizontal, el punto P y la cámara, estos lados son x_0 y z_0, mediante el teorema de Pitágoras se obtendrá la hipotenusa c_0 que forma parte del segundo triángulo formado por la trayectoria del láser.

$${c_0}^2 = {z_0}^2 + {x_0}^2 \tag{17}$$

Despejando c_0 mediante la raíz cuadrada:

$$c_0 = \sqrt{{Z_0}^2 + {x_0}^2} \tag{18}$$

Se aplica la ley de cosenos en el nuevo triángulo para obtener el valor a_0, el único lado del triángulo que aún no se conoce.

$$a_0 = \sqrt{b^2 + {c_0}^2 - 2 \cdot b \cdot c_0 \cdot cos(A_0)} \tag{19}$$

Por último, los únicos valores desconocidos del triángulo son los ángulos B_0 y C, el ángulo C será una constante para todo el sistema ya que el ángulo de incidencia del láser es permanente para todo el montaje. Se obtienen a partir de la ley de senos, ecuación (20).

$$\frac{a_0}{\sin(A_0)} = \frac{b}{\sin(B_0)} = \frac{c_0}{\sin(C)} \tag{20}$$

La ecuación (20) permite despejar los valores B_0 y C, en función de la recta a_0 y el ángulo A_0:

$$B_0 = \sin^{-1}\left[\frac{\sin(A_0)}{a_0} \cdot b\right] \tag{21}$$

$$C = \sin^{-1}\left[\frac{\sin(A_0)}{a_0} \cdot c_0\right] \tag{22}$$

Con este último ángulo se tiene completamente caracterizado el triángulo generado, por lo que todas las relaciones laser-pared-cámara quedan totalmente caracterizadas, al conocerse x_0, a_0, c_0, A_0, B_0 y C.

2.2.9 Calculo de relaciones con el objeto

A continuación, se estudiará en profundidad el impacto del láser con el plano del objeto, analizando el triángulo que forma individualmente, obviando el análisis anterior y exponiendo únicamente este triángulo. Dicho triángulo está representado en la Fig. 13. El objetivo final es obtener el valor de distancia al plano del objeto (z_1).

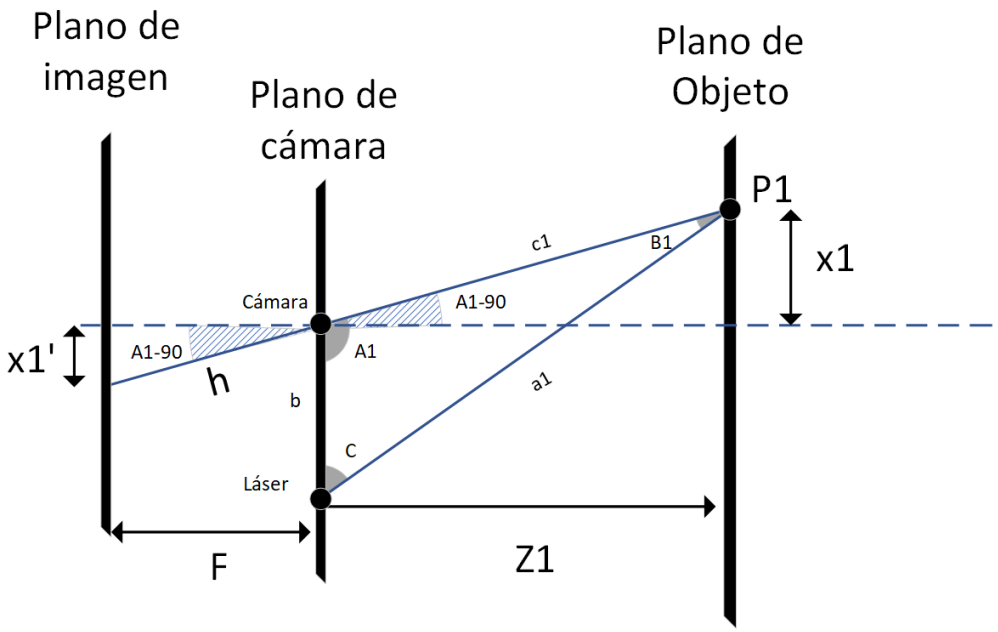


Fig. 13. Análisis de relaciones trigonométricas entre la tripla láser-cámara-objeto.

Al igual que en la caracterización anterior del triángulo se comienza aplicando las propiedades de la cámara *pinhole* para obtener el ángulo de incidencia a la cámara A_1. Se plantean ambas ecuaciones:

$$x_1' = h \cdot sin(A_1 - 90) \tag{23}$$

$$F = h \cdot cos(A_1 - 90) \tag{24}$$

A partir de estas ecuaciones se llega a la ecuación (25):

$$\tan(A_1 - 90) = \frac{x_1'}{F} \tag{25}$$

Mediante aplicar la función arcotangente en ambos lados de la ecuación (25) se llega a la fórmula (26) que despeja el valor del ángulo A_1.

$$A_1 = 90 + \tan^{-1}\left(\frac{x_1'}{F}\right) \tag{26}$$

Conocidos los ángulos A_1 y C (obtenido en el análisis Cámara-Pared), es posible obtener B_1 de forma directa, utilizando las propiedades de los triángulos:

$$A_1 + B_1 + C = 180° \tag{27}$$

Despejando el ángulo B_1:

$$B_1 = 180° - (A_1 + C) \tag{28}$$

Con el ángulo B_1 obtenido se procederá a aplicar la ley de senos para obtener el lado c_1 con el objetivo de obtener su proyección en el eje X, determinando asi la distancia entre la cámara y el objeto.

$$\frac{a_1}{\sin(A_1)} = \frac{b}{\sin(B_1)} = \frac{c_1}{\sin(C)} \tag{29}$$

Mediante la fórmula (25), se llega a calcular la expresión (26) que despeja el valor c_1:

$$c_1 = \frac{\sin(C)}{\sin(B_1)} \cdot b \tag{30}$$

Se aplica la definición de coseno con el objetivo de obtener la distancia real entre el plano del objeto y el plano de cámara, dicha definición parte del triángulo representado en la Fig. 14.

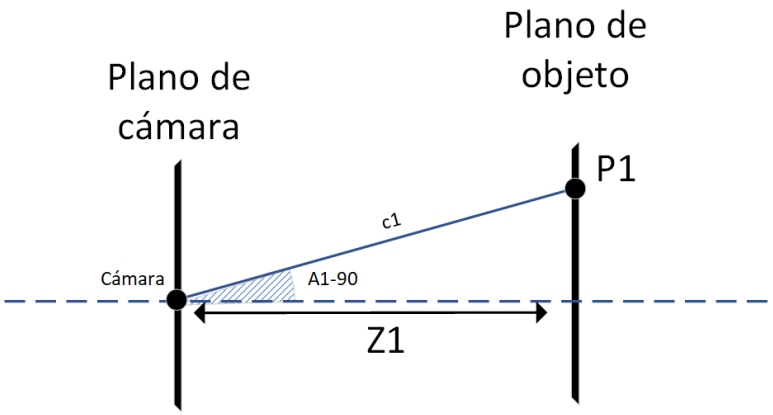


Fig. 14. Obtención de la distancia z_1 entre ambos planos.

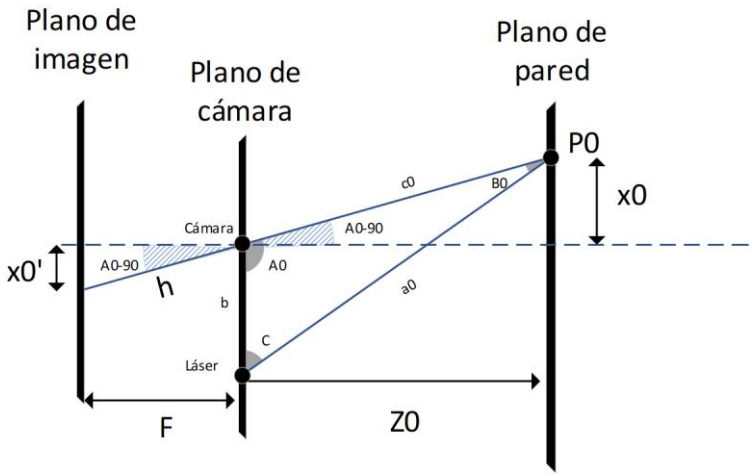
Conocidos los valores de A_1 y c_1, se obtiene la distancia Z_1, único objetivo de este apartado mediante la ecuación (31):

$$Z_1 = c_1 \cdot \cos(A_1 - 90) \tag{31}$$

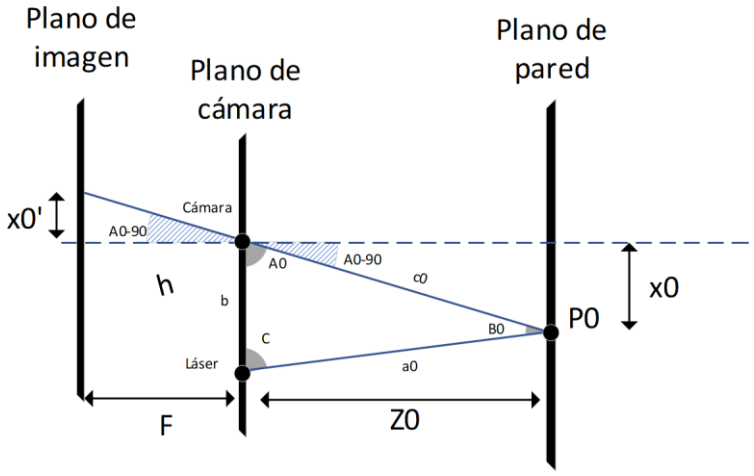
Las relaciones laser-objeto-cámara quedan completamente caracterizadas, al conocerse los ángulos A_1, B_1, c y las distancias x_1, c_1 y Z_1. Además, mediante la expresión (31) se obtiene la distancia en el eje Z entre el plano cámara-lente y el objeto, Z_1. Este dato proporciona la información sobre la superficie del objeto, necesaria para obtener su vista 3D.

2.2.10 Función arcotangente

A continuación, se demostrará de la validez de la ecuación (15) para ángulos mayores de 90º y para ángulos menores de 90º. Por lo que se generalizará la expresión para ángulos de cámara de entre (0, 180)º. Se representan ambos ángulos de entrada en la Fig. 15.



(a)



(b)

Fig. 15. Comparación de relaciones trigonométricas para ángulos de cámara mayores de 90° (a) y menores de 90° (b).

Se despeja el ángulo A_0 en ambas situaciones, para el ángulo mayor de 90º igual que en la situación resuelta con anterioridad y en el caso del ángulo menor de 90º se invierte el triángulo respecto al eje horizontal y se opera con el complementario de dicho ángulo.

$$x_0' = h \cdot sin(A_0 - 90) \qquad x_0' = h \cdot sin(90 - A_0) \tag{32, 33}$$

$$F = h \cdot cos(A_0 - 90) \qquad F = h \cdot cos(90 - A_0) \tag{34, 35}$$

Se relacionan ambas ecuaciones y se despeja A_0:

$$\tan(A_0 - 90) = \frac{x_0'}{F} \qquad \tan(90 - A_0) = \frac{x_0'}{F} \tag{36, 37}$$

$$A_0 = 90 + \text{atan}\left(\frac{x_0'}{F}\right) \qquad A_0 = 90 - \text{atan}\left(\frac{x_0'}{F}\right) \tag{38, 39}$$

Resumiendo, el símbolo del arcotangente se verá modificado con respecto al tipo de ángulo A_0 del que se disponga. Para solucionar esta situación se ha realizado la indexación inicial del mapa de pixeles, cuando el ángulo alcance el umbral de los 90º hacia valores positivos el valor de x_0' será un valor positivo, de igual forma al descender hacia valores menores de 90º el valor de x_0' se tornará en valores negativos.

Se observa, en la Fig. 16, la forma de la función arcotangente simulada en Matlab®.

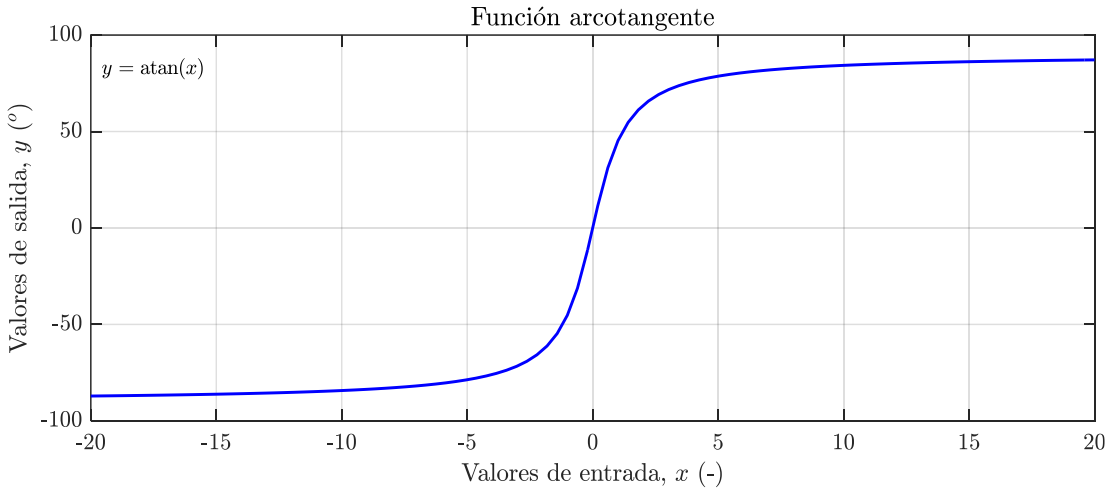


Fig. 16. Respuesta de la función arcotangente.

Nos encontramos ante una función con simetría impar, es decir, una función que presenta una simetría central respecto al origen de coordenadas. Por lo que se verifica:

$$\text{atan}\,(x) = -\text{atan}\,(-x) \tag{40}$$

Para el ángulo mayor de 90º los valores de x_0' continúan positivos por lo que la función permanece inmutable, los valores siempre serán positivos.

$$A_0 = 90 + \text{atan}\left(\frac{x_0'}{F}\right) \qquad x_0' > 0 \tag{41}$$

Se introduce el valor absoluto:

$$A_0 = 90 + \operatorname{atan}\left(\frac{|x_0'|}{F}\right) \tag{42}$$

Para el ángulo menos de 90º los valores de x_0' son negativos, por lo que aplicamos la simetría impar de la función arcotangente para poder generalizar la expresión para ambas situaciones.

$$A_0 = 90 - \operatorname{atan}\left(\frac{x_0'}{F}\right) \qquad x_0' < 0 \tag{43}$$

$$A_0 = 90 - \left[-\operatorname{atan}\left(\frac{-x_0'}{F}\right)\right] \qquad x_0' < 0 \tag{44}$$

Se introduce el valor absoluto:

$$A_0 = 90 - \left[-\operatorname{atan}\left(\frac{|x_0'|}{F}\right)\right] \tag{45}$$

$$A_0 = 90 + \operatorname{atan}\left(\frac{|x_0'|}{F}\right) \tag{46}$$

Se llega a la misma expresión en la cual se opera indistintamente la distancia al eje vertical de cada uno de los puntos, sin verse modificada la expresión por el tipo de ángulo que forma.

2.2.11 Plataforma de giro

Con la finalidad de posicionar el objeto a escanear a una distancia fija de la cámara y el láser, se ha diseñado y fabricado una plataforma cilíndrica, de radio 6,5 cm y altura 0,5 cm. Adicionalmente, esta plataforma debe facilitar el giro del objeto a escanear sobre el eje Y del sistema de coordenadas definido en la Fig. 8, por lo que continua una base rotatoria físicamente conectada al motor paso a paso.

La plataforma de giro ha sido diseñada con el software AutoCAD 2021® y Ultimaker Cura® 8.5.1, y fabricado con la impresora Peopoly Moai 130® mostrada en la Fig. 17. Esta impresora 3D realiza impresiones por estereolitografía, basada en el endurecimiento de resina líquida depositada en su tanque, a través de la aplicación de una fuente de luz. Las resinas SLA (materiales termo-endurecibles reactivos a la luz) responden a ciertas longitudes de onda, solidificándose y transformándose en plástico endurecido. Los objetos 3D se construyen por un proceso aditivo de capas de resina endurecida y adheridas de forma consecutiva. Sus principales ventajas respecto a las impresoras de Modelado por Deposición Fundida (MDF) son una mayor resolución y precisión, unida a una terminación superficial más lisa.

Fig. 17. Imagen de impresora 3D, modelo Poepoly Moai 130 [9].

Tal y como puede apreciarse en los planos de la plataforma mostrados en la Ilustración 17, esta se compone de una estructura cilíndrica principal de diámetro $12\ cm$ y altura $6{,}5\ cm$. La estructura presenta unas paredes de grosor 1,5 cm, y una cavidad interna diseñada para fijar el motor paso a paso. Finalmente, el cilindro principal este cubierto por una plataforma rotatoria de idéntico diámetro y altura 1 cm.

Como se ha comentado, el interior del cilindro ha sido diseñado para fijar la posición de motor paso a paso, por lo que cuenta con una estructura de base cuadrada de $4 \times 4 \times 3{,}2\ \text{cm}^3$ donde posicionar el motor, además de unas pestañas de sujeción que impiden su desplazamiento. Además, como se observa en la Fig. 18 el cilindro presenta un agujero lateral de radio 0,5 cm, utilizado para facilitar el cableado de motor. Una vez introducido el motor en la base inferior del cilindro, se debe insertar su rotor en la base rotatoria, para realizar la transferencia de movimiento. Para este fin, como se aprecia en la Fig. 18, la base rotatoria cuenta con una abertura central diseñada para encajar y fijar el rotor. Debe considerarse que el motor paso a paso presenta un rotor cilíndrico, sin ningún tipo de arista que facilita su fijación a la base rotatoria. Por tanto, ha sido necesario modificar manualmente la sección del rotor, de forma que se asegure una fijación estable con la base y se garantice la transferencia del movimiento de forma precisa. La Ilustración 20 incluye el detalle de la modificación realizada al eje (rotor) del motor, de forma que en sección original circular ha pasado a tener forma semicircular. De forma análoga, la abertura central de la base rotatoria presenta una sección de la misma forma y dimensiones, lo que asegura el conexionado de ambos elementos y la transferencia fiable del movimiento.

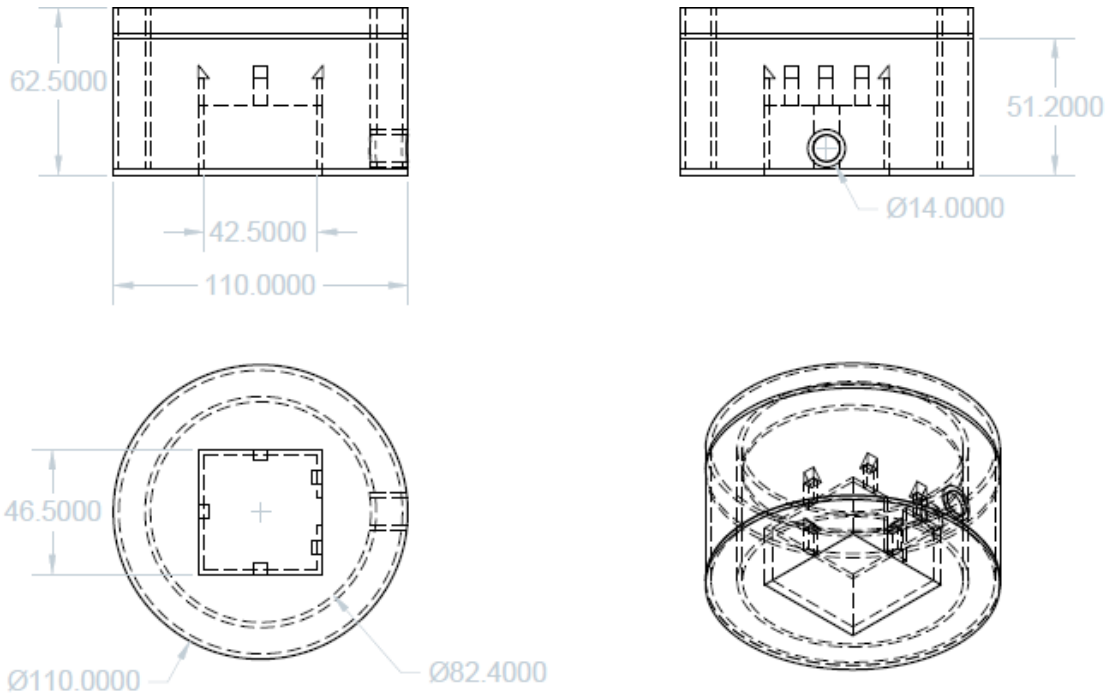


Fig. 18. Planos de la estructura cilíndrica de soporte.

El diseño de la base superior se ha visto modificado con la implementación de una malla hexagonal con el objetivo de disminuir su peso, y no oponer resistencia al giro del rotor, dicha malla es apreciable en los planos de dicha plataforma, Fig. 19.

Con el fin de lograr una mayor fijación de los componentes se introduce una unión de plástico con la forma del rotor modificado con un semicírculo en su punta, dicha unión se ve representada en la Fig. 20.

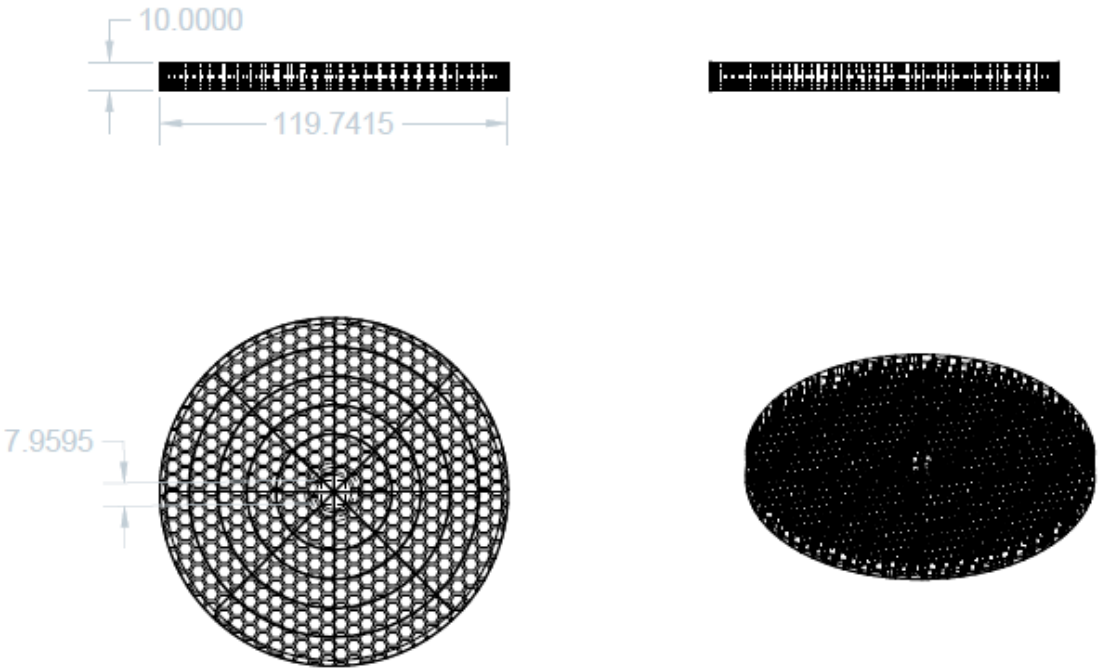


Fig. 19. Planos de la plataforma rotatoria superior.

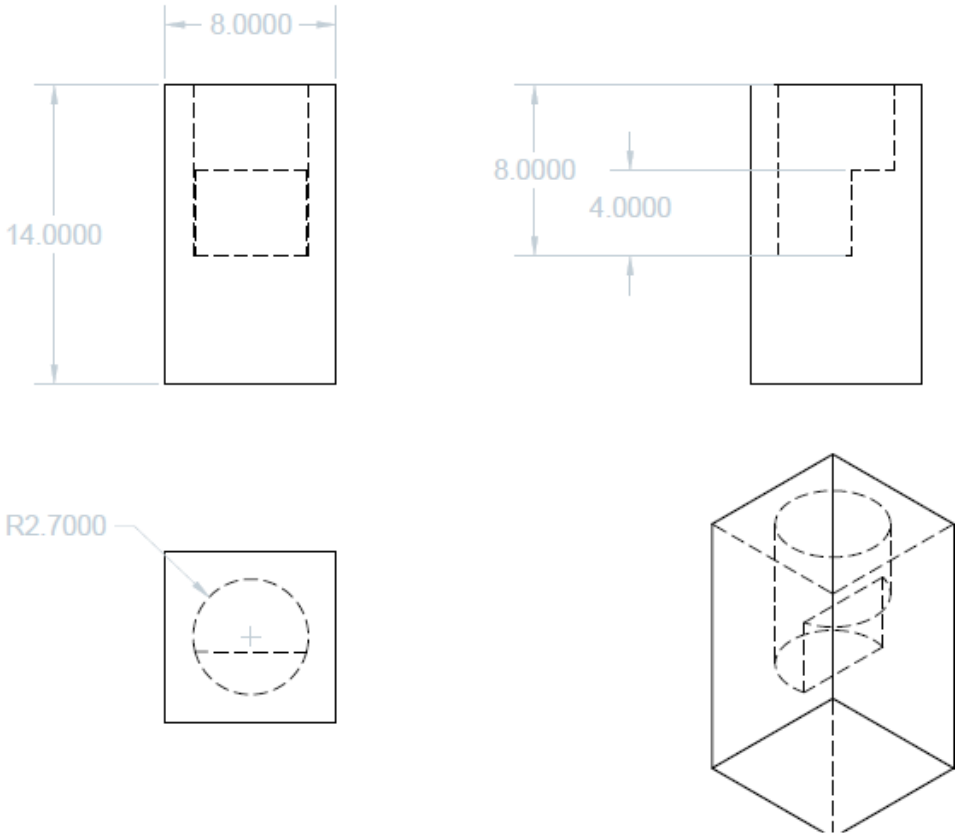


Fig. 20. Planos de la pieza de conexión entre el rotor y la plataforma superior.

Para terminar, la base cilíndrica incluye una canalización en su extremo superior, en la que, como se aprecia en la Fig. 21, se han añadido rodamientos. Así, la base rotatoria descansará sobre los rodamientos, lo que minimizará la fricción de la base y facilitará una rotación homogénea. La base superior encaja encima de dichos rodamientos como se aprecia en la Fig. 22.

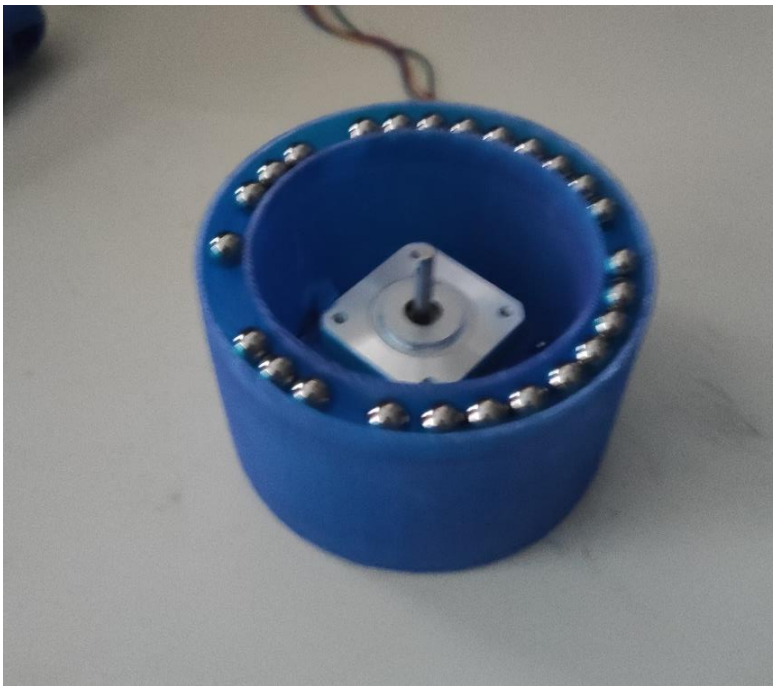


Fig. 21. Fotografía de la estructura cilíndrica sin la plataforma superior rotatoria.

Fig. 22. Fotografía de la estructura cilíndrica con la plataforma superior rotatoria.

2.2.12 Control de giro de precisión

En este apartado se describe en detalle el algoritmo de control de giro del motor, el cual está condicionado por las características del hardware utilizado. En este sentido, el control del motor se realiza a través de la placa Arduino Uno®, la cual genera señales de salida de baja tensión, limitadas a 5 V. Sin embargo, el motor bipolar paso a paso requiere de mayores tensiones, generalmente 12 V, además de intensidades de hasta 2,5 A, para operar correctamente. Por esta razón, el sistema requiere de un bloque microcontrolador de motor (DRV825), encargado de transformar las señales de control generadas por Arduino® en señales de valores apropiados para inducir cambios en el estado del motor.

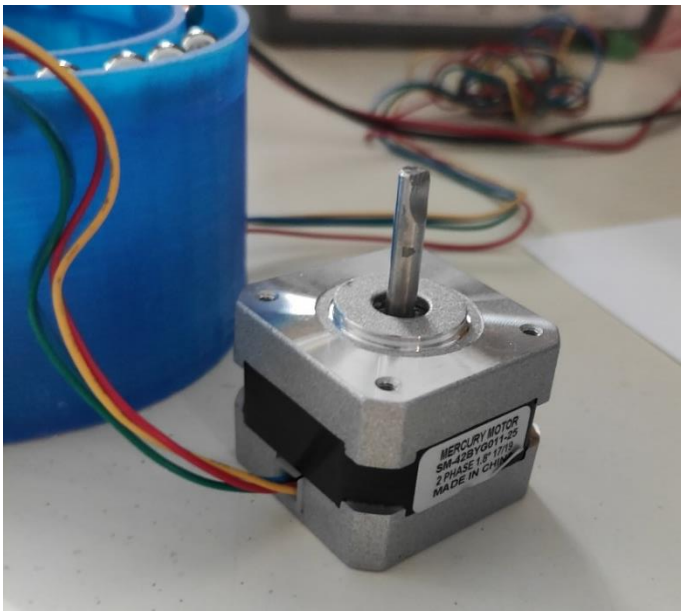


Fig. 23. Fotografía de motor paso a paso modificado.

El motor bipolar paso a paso utilizado puede definirse como un dispositivo electromecánico capaz de generar desplazamientos angulares de su eje (rotor) a partir de la aplicación de pulsos eléctricos. De forma sencilla, como se muestras en la Fig. 23, un motor paso a paso bipolar de imán permanente presenta dos partes diferenciadas: el rotor y el estator. En este caso, el rotor puede aproximarse por un eje de rotación dotado de uno o varios imanes permanentes. A su vez, el estator está formado por dos bobinas fijas en un ángulo de 90º, junto a las que se ubica el rotor. Cada terminal de las bobinas está conectado a un cable de entrada/salida independiente, lo que permite inyectar corrientes en ambos sentidos, cambiando la polaridad del campo magnético generado por la inductancia (solenoide). Dependiendo de la polaridad del campo magnético. El rotor imantado se verá atraído y/o repelido, generándose una rotación controlada del eje del motor. Por tanto, el giro completo del rotor estará controlado por el sentido de la corriente eléctrica inyectada a las bobinas del estator [10].

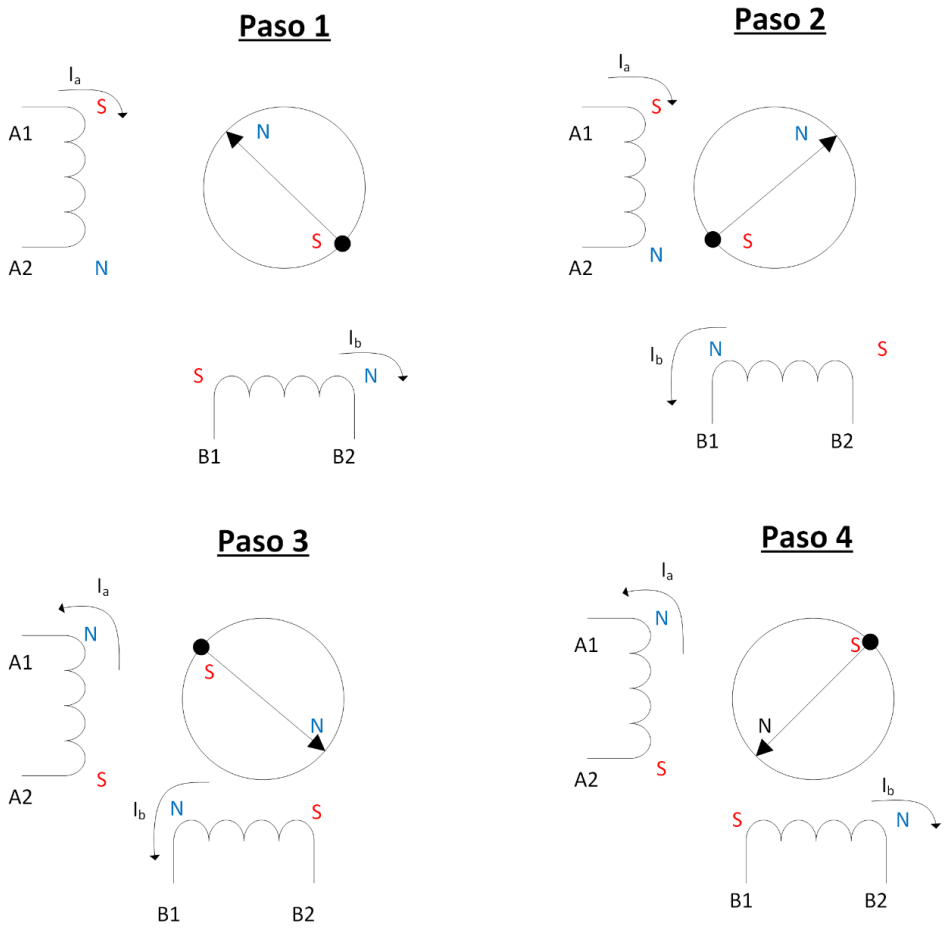


Fig. 24. Secuencia de entradas para completar un giro del motor en pasos de 90º.

Paso	A1	A2	B1	B2
I	+	-	+	-
II	+	-	-	+
III	-	+	-	+
IV	-	+	+	-

Tabla 4. Secuencia de tensiones de entrada para completar un giro del motor en pasos de 90º.

A modo de ejemplo, Fig. 24 muestra la secuencia de entradas (tensiones de entrada) requerida en los terminales de las bobinas para completar un giro del rotor en cuatro pasos de 90º cada uno, las tensiones de control han de ser semejantes a las representadas en la Tabla 4.

El microcontrolador utilizado es el DRV8825 de Texas Instruments, cuya imagen se muestra en la Fig. 25, y cuya hoja de características se incluye en el Anexo I de este documento, como se ha comentado, este microcontrolador tiene la función de adaptar las señales de control de Arduino® a niveles compatibles con los requeridos por el motor. Además, el microcontrolador DRV8825 será el encargado de proporcionar al motor la secuencia de pulsos de entrada, que permita polarizar sus bobinas en el orden adecuado, para obtener un giro del rotor con el sentido y precisión deseados. Para esta finalidad, el controlador dispone de dos circuitos de tipo puente H, que permite inducir una corriente eléctrica en cada bobina con el sentido deseado, tal y como se puede reconocer en la Fig. 26.

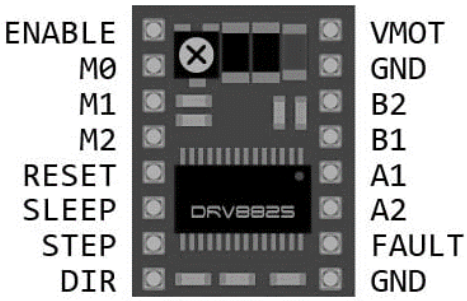


Fig. 25. Imagen del chip controlador DRV8825 [6].

Un puente H es un circuito electrónico que permite a un motor eléctrico de corriente continua girar en avance y retroceso. Se forman cuatro interruptores, mediante la integración de transistores, en una estructura con forma de H que tiene como centro una de las bobinas del motor. Además del uso de transistores el sistema se acompaña habitualmente de diodos inversamente polarizados con el objetivo de proteger el sistema de picos transitorios que generan las bobinas. Al encontrarse en un sistema con motor bipolar se hacen uso de dos puentes H, uno para cada bobina.

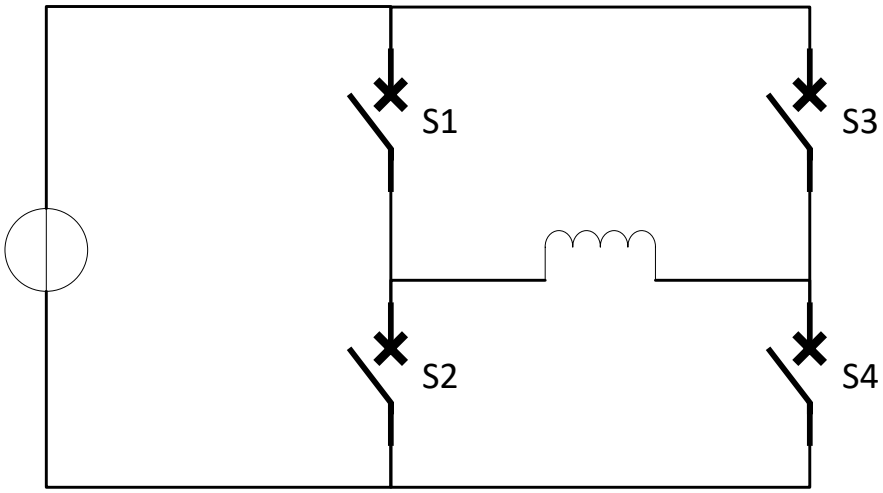


Fig. 26. Esquemático básico del circuito de puente H.

El puente H permite forzar el giro del motor en ambos sentidos, pero también puede utilizarse para frenarlo al realizar un cortocircuito entre las bornas de la bobina, o incluso para conseguir que el motor gire bajo su propia inercia, cuando se desconecta completamente su alimentación. A modo de resumen se indica en la Tabla 5 todas las posibles configuraciones de giro del motor permitidas por los estados del puente H.

	Interruptor S1	Interruptor S2	Interruptor S3	Interruptor S4
Giro Avance	ON	OFF	OFF	ON
Giro Retroceso	OFF	ON	ON	OFF
Inercia	OFF	OFF	OFF	OFF
Estado de freno 1	ON	OFF	ON	OFF
Estado de freno 2	OFF	ON	OFF	ON
Cortocircuito de la fuente (estado a evitar)	ON	ON	OFF	OFF
	OFF	OFF	OFF	OFF
	ON	ON	ON	ON

Tabla 5. Estados del puente H

Por otro lado, una característica importante del micocontrolador es la posibilidad de realizar un indexado *microstepping*. Típicamente un motor paso a paso cuenta con 200 pasos o posiciones, por lo que se inducen 1.8 grados al giro de la plataforma por cada paso. Sin embargo, con el sistema de indexado *microstepping* es posible realizar pasos de tan solo 0.05625º, necesitándose 6400 pasos para completar un giro.

Para lograr este indexado, el sistema regula la cantidad de corriente que transmite cada interruptor. En el procedimiento original, cada interruptor se comporta de forma binaria dejando entrar de forma brusca toda la corriente a la bobina. En este procedimiento de *microstepping*, el sistema emulara un sistema analógico donde el progreso de cada paso se construye como si se tratase de una señal sinusoidal. Como es evidente, la señal no es una sinusoide perfecta, sino que se realizara de forma digital con pequeños escalones, cada uno de ellos denominado un *microstep*. Este tipo de funcionamiento del microcontrolador se detalla dentro de sus hojas de características, donde se incluye la Fig. 27, mostrando las señales de polarización de las bobinas del motor por *microstepping*.

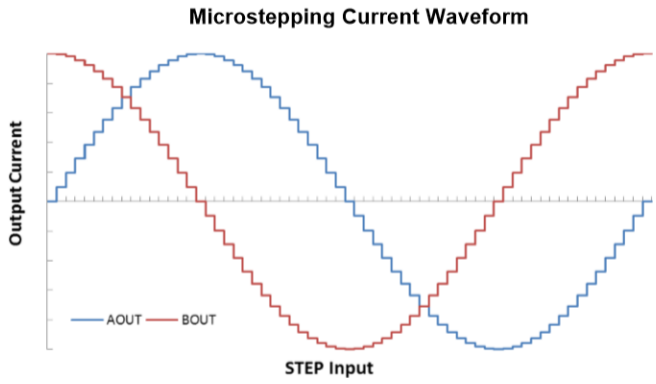


Fig. 27. Forma de onda de señales de alimentación de bobinas para *microstepping*.

Una vez introducido el funcionamiento del microcontrolador se deben considerar sus requerimientos de operación. Para manejar el controlador solo se necesitan dos pines, uno para la dirección de giro (DIR) y otro para avanzar un paso ($STEP$). Por otra parte, la cantidad de pasos se configura con los pines "M0","M1" y "M2".

La siguiente tabla del $datasheet$ ilustra las distintas configuraciones:

	Pasos	Pin M2	Pin M1	Pin M0
Full Step	200	OFF	OFF	OFF
1/2 Step	400	OFF	OFF	ON
1/4 Step	800	OFF	ON	OFF
1/8 Step	1600	OFF	ON	ON
1/16 Step	3200	ON	OFF	OFF
1/32 Step	6400	ON	OFF	ON
		ON	ON	OFF
		ON	ON	ON

Tabla 6. Listado de configuraciones de pasos del motor.

La señal de activación ($nENBLE$) posibilita el funcionamiento de todo el microcontrolador, se trata de una señal negada por lo que deberá permanecer en valor bajo durante toda la ejecución. Otras señales para tener en cuenta serán los pines de reinicio ($nRESET$) y de $sleep$ ($nSLEEP$) que permanecerán en valor alto durante la ejecución normal del sistema. La señal $nRESET$ se encarga de reiniciar el motor cuando toma un valor bajo. De forma similar la señal $nSLEEP$ se encarga de poner en espera al motor cuando toma un valor bajo, de forma que el dispositivo no presente consumo de potencia en los periodos en los que se encuentre inactivo. Una descripción más detallada de la funcionalidad de la señal $nSLEEP$ se incluye en la parte final de este apartado, remarcando su importancia para controlar el consumo y disipación del calor del motor. Por último, los pines de alimentación del motor a 12 V ($VMOT$) y alimentación del controlador a 5 V (VDD) así como dos tierras (GND), una para cada tensión de alimentación.

La señal $STEP$, que provoca el desplazamiento del motor a su próxima posición tiene ciertos requerimientos temporales que se deben asumir y se explican con detalle a continuación. La señal $STEP$ ha de estar compuesta por un pulso de 1,9 μs de subida seguido de un pulso de 1,9 μs de bajada. Las señales de control y la señal de cambio de dirección se han de monitorizar para el correcto funcionamiento del controlador, según se aprecia en la Ilustración 25.

Por lo tanto, tras un cambio de dirección en la señal DIR se deberá evitar realizar un pulso $STEP$ durante 650 ns y deberá permanecer en esa dirección por otros 650 ns tras la subida del pulso. Así mismo, la activación de la señal $nSLEEP$ debe de respetarse con un delay de 1,7 ms antes del siguiente pulso $STEP$. Por último, el tiempo de espera para un pulso $STEP$ que se ha de respetar ante un flanco de bajada en la señal $nENBL$ es de 650 ns.

			MIN	MAX	UNIT
1	f_{STEP}	Step frequency		250	kHz
2	$t_{WH(STEP)}$	Pulse duration, STEP high	1.9		µs
3	$t_{WL(STEP)}$	Pulse duration, STEP low	1.9		µs
4	$t_{SU(STEP)}$	Setup time, command before STEP rising	650		ns
5	$t_{H(STEP)}$	Hold time, command after STEP rising	650		ns
6	t_{ENBL}	Enable time, nENBL active to STEP	650		ns
7	t_{WAKE}	Wakeup time, nSLEEP inactive high to STEP input accepted		1.7	ms

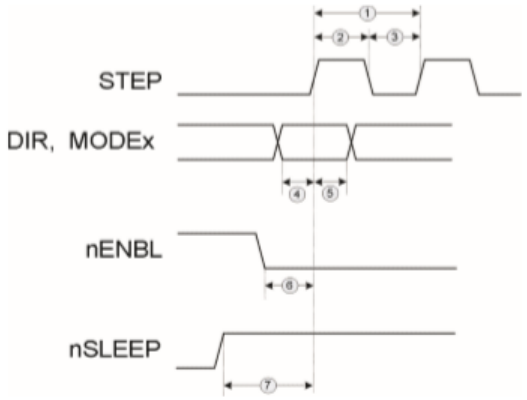


Fig. 28. Esquema de los requerimientos temporales en las señales de control del microcontrolador

Como es de esperar, el programa de control de motor en Arduino® ha sido diseñado respetando los requerimientos temporales indicados en la Fig. 28, forzando retrasos (paradas) en el código cuando se han requerido para realizar la secuencia de señales de la forma indicada.

Por último, se encuentran las señales de control/polarización de cada par de bobinas (A1, A2, B1, B2), las cuales se conectan directamente al motor. El conexionado del microcontrolador con el motor se realiza siguiendo las indicaciones de sus hojas de características [6], donde se indican los bornes de conexión de cada bobina con un código de colores, mostrado en la Fig. 29, de forma que las señales A1 y A2 se conectan a los cables rojo y verde respectivamente para polarizar la primera de las bobinas. De forma similar, la segunda bobina se polariza a través de las señales B1 y B2, conectadas a los cables amarillo y azul, respectivamente.

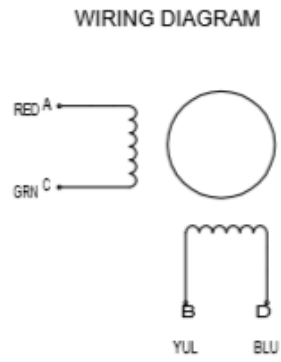


Fig. 29. Esquemático interno del motor paso a paso simplificado.

Finalmente, es necesario indicar que, durante la comprobación del sistema de giro, se hizo patente el consumo del motor cuando este se encontraba inmóvil. Este consumo es debido a que la intensidad circulando por las bobinas no es nula cuando el motor está parado, lo que implica un consumo constante de potencia. Debido a esta razón, se observa la necesidad de utilizar alguna

herramienta capaz de limitar el flujo de corriente por las bobinas del motor cuando este no presenta movimiento.

Al analizar las funciones del controlador, se identifica una función $SLEEP$ que se acciona con la entrada de un valor bajo en el pin $nSLEEP$. Esta función deshabilita todos los interruptores de los puentes H. Al encontrarse en esta situación, el motor entra en el estado de inercia previamente comentado, donde ningún campo magnético regula la posición del rotor. Además, en este estado $SLEEP$ se deshabilita la entrada de corriente del pin conectado a 12 V limitándose el consumo de potencia del motor y su excesivo calentamiento. Adicionalmente, la habilitación de la señal $SLEEP$ provoca la parada de los relojes internos del microcontrolador.

2.2.13 Comunicación Matlab® Arduino®

Para realizar la comunicación entre las plataformas se implementa un conjunto de instrucciones con confirmación de ejecución por parte de Arduino®. Como se ha comentado la base de esta comunicación es el puerto serie de la placa Arduino Uno® y el ordenador con el software Matlab®.

La estructura final de la comunicación es la que se aprecia en la Fig. 30, donde los bloques de ejecución grises son los pertenecientes a la plataforma Arduino® y los bloques azules representan los fragmentos de ejecución del script Matlab®. Para llevar a cabo esta estructura se ha decidido llevar a cabo una comunicación con intercambio de relevo. Es decir, mientras una plataforma ejecuta su código, la otra plataforma permanece en espera hasta recibir el relevo.

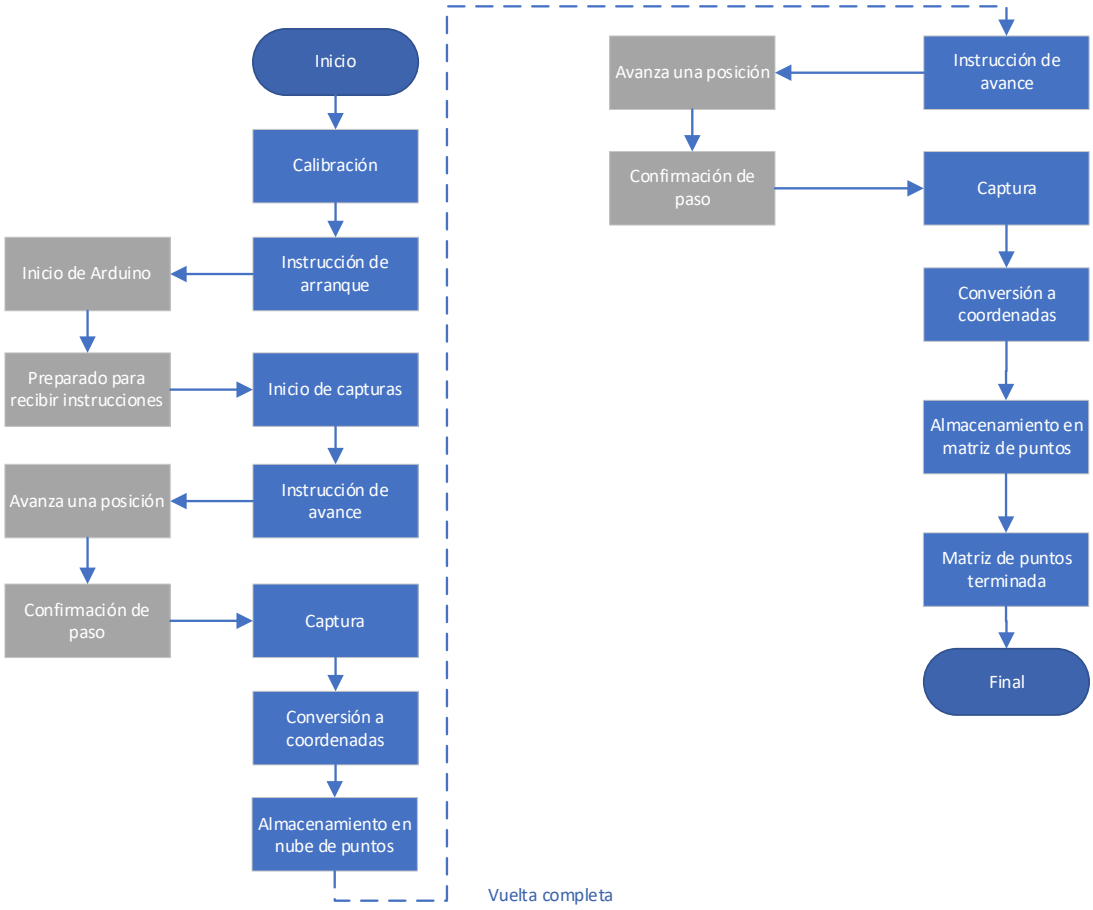


Fig. 30. Esquema del flujo de operación que se realizará en el algoritmo.

La placa Arduino Uno® cuenta con una unidad UART (Universally Asynchronous Receiver/Transmitter), encargada de realizar la conversión de los datos a una secuencia de bits y transmitirlos/recibirlos a una determinada velocidad. Esta unidad en concreto opera a un nivel TTL (Transistor-Transistor Logic) de 5V lo que permite una compatibilidad completa con la conexión USB estándar.

Se configura la conexión a la máxima velocidad permitida por la placa Arduino®, fijándose a un valor de 115200 baudios. La tasa de baudios establece el número de unidades de símbolos transmitidos por segundo. Para evitar posibles limitaciones del sistema, se ha establecido la velocidad al máximo permitido, de forma que el tiempo entre el inicio y final de una transmisión sea despreciable frente al tiempo requerido por las instrucciones ejecutadas en la placa Arduino y en el software Matlab®.

A continuación, se establece un terminador del mensaje con un retorno de carro (CR/LF). Cuando el puerto serie encuentre un retorno de carro en el mensaje se procederá a enviar todo el stack de salida. Gracias a esta estructura el tamaño de los mensajes podrá ser variable, permitiendo a la placa Arduino® enviar y recibir cadenas de caracteres de cualquier tamaño.

El conjunto de instrucciones desarrolladas para controlar el funcionamiento de la placa Arduino Uno® desde el software Matlab® se recopila en la Tabla 7. Así pues, siempre que se envíe a través del puerto serie una de las cadenas de caracteres indicadas en la Tabla 7, seguidas de un carácter de retorno de carro, la placa Arduino ejecutará el fragmento de código asociado a dicha instrucción. Puede decirse que la programación de Arduino implementada gira en torno a esta comunicación, ya que la rutina principal comprueba continuamente el $buffer$ de entrada hasta que detecta la presencia de instrucciones.

Cadena de caracteres	**Función**	**Descripción**	**Mensaje de Confirmación de Arduino**
"encenderMotor"	encenderMotor()	Activa los pines enable del controlador	"Motor Encendido"
"apagarMotor"	apagarMotor()	Desactiva los pines enable del controlador	"Motor Apagado"
"unPaso"	un_paso()	Envía un pulso de activación al controlador	"PasoDado"
"200Pasos"	pasos(200)	Activa los pines necesarios para la configuración 200 pasos	"200 Pasos"
"400Pasos"	pasos(400)	Activa los pines necesarios para la configuración 200 pasos	"400 Pasos"
"800Pasos"	pasos(800)	Activa los pines necesarios para la configuración 200 pasos	"800 Pasos"
"1600Pasos"	pasos(1600)	Activa los pines necesarios para la configuración 200 pasos	"1600 Pasos"
"3200Pasos"	pasos(3200)	Activa los pines necesarios para la configuración 200 pasos	"3200 Pasos"
"6400Pasos"	pasos(6400)	Activa los pines necesarios para la configuración 200 pasos	"6400 Pasos"

"cambioGiro"	cambioGiro()	Envía un pulso de cambio de sentido de giro al controlador	"Sentido de giro cambiado"
"cambioLaser"	cambioLaser()	Modifica el interruptor del láser on/off	"Estado de láser modificado"

Tabla 7. Tabla de instrucciones

Por tanto, el paso inicial de la comunicación Matlab®-Arduino® consiste en la configuración del puerto serie. Por defecto, la placa Arduino está en continua comprobación del puerto serie, presentando un tiempo de espera predeterminado de 1 s para refrescar el contenido del $buffer$. Con el objetivo de reducir este tiempo, la conexión se configura con un tiempo de espera de 10 ms. Así ante la falta de detección de un terminador retorno de carro en el $buffer$ de entrada, la espera máxima para la ejecución de una instrucción se verá limitada a 10 ms.

Una vez configurada la conexión entre Matlab® y Arduino®, es posible comenzar con el protocolo de comunicación desarrollado. La estructura de la comunicación se muestra en la figura 28 y consta de los siguientes pasos principales:

I) Inicio del sistema y configuraciones de conexión con tiempo de espera 10 ms y tasa de transmisión de 115200 baudios. El sistema se sincroniza vía puerto serie mediante un $reset$ de la plataforma Arduino®. Tras este $reset$ el script Matlab® permanece a la espera del envío de una cadena de caracteres "ACK" por parte de Arduino® para confirmar el inicio.

II) La plataforma Arduino® configura sus salidas de forma predeterminada con una configuración de giro en sentido horario y un stepping de 200 pasos. Una vez configurado, Arduino envía la cadena "ACK" a modo de confirmación y entra en modo de escucha; leyendo el $buffer$ de entrada cada 10 ms o cada vez que detecte un carácter de retorno de carro.

III) Tras este paso, Arduino espera la llegada de una instrucción, tras la cual ejecuta el fragmento de código seleccionado y retorna una cadena de caracteres a modo de testigo. Durante la ejecución de este fragmento de código, el script Matlab® permanece a la espera.

IV) Por último, tras enviar la confirmación de envío, el código de Arduino retorna a su modo de escucha, y lee el $buffer$ de entrada hasta la llegada de una nueva instrucción por parte de Matlab.

En el siguiente esquema de la Fig. 31 se aprecia la estructura de la comunicación en un ejemplo real:

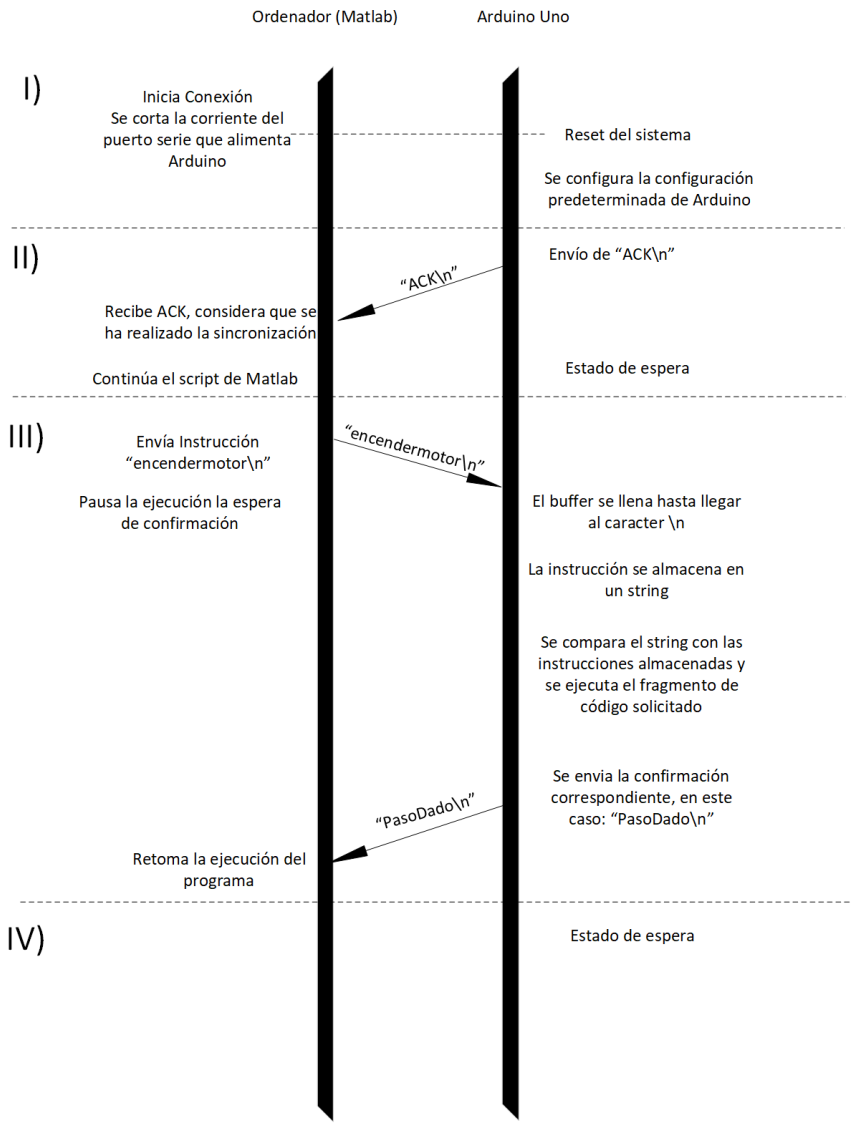


Fig. 31. Esquema de la comunicación típica entre las plataformas Matlab® y Arduino®.

Esta estructura se utiliza de forma continuada en cada iteración del proceso logrando obtener una imagen por cada paso realizado. Así, es posible conseguir una sincronización entre giros y capturas con el objetivo de evitar la repetición de capturas innecesarias para el modelado tridimensional.

2.2.14Detección láser

La obtención de la fotografía del objeto iluminado por el láser, su análisis para la obtención de la distancia de la superficie iluminada respecto al plano láser-cámara, supone la funcionalidad principal del bloque de procesado de datos.

Así pues, con la finalidad de obtener una representación digital en 3D del objeto de interés, se procederá al procesado de la imagen, como la imagen de la Fig. 32, almacenada en una matriz de dimensiones $1280 \times 720 \times 3$. El objetivo final de este procesado es el conseguir una matriz binaria de dimensiones 1280×720 que represente la posición horizontal de cada punto de luz láser, es decir, se filtre la dispersión de luz detectando un único punto de incidencia por cada fila.

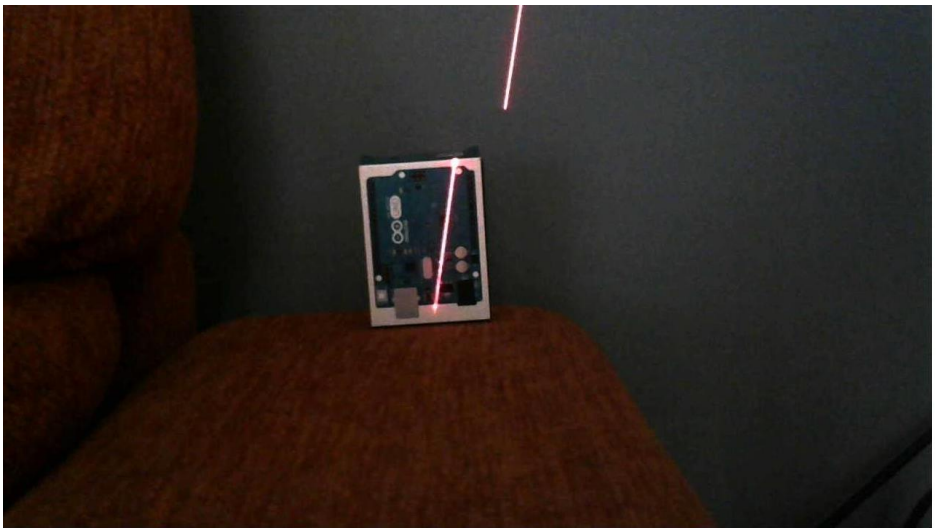


Fig. 32. Fotografía utilizada como entrada "FrameOn" del algoritmo.

1) Filtro RGB

El primer paso del procesado de la imagen consiste en la aplicación de un filtro de componentes RGB. En este sentido, es necesario recordar que la imagen capturada presenta tres componentes de color para cada píxel, correspondientes a las longitudes de onda del rojo (R), verde (G) y azul (B), codificando la intensidad de cada uno de ellos en un rango de [0,255], donde el valor 0 indica la ausencia de dicho color en el píxel, mientras que un valor de 255 implica la mayor intensidad posible de ese color en el píxel.

El láser utilizado en el sistema presenta una longitud de onda característica de $\lambda = 643$ nm, correspondiente al color rojo del espectro de luz visible. El filtro RGB a aplicar persigue modificar la matriz de la imagen, imagen $(x, y)_{RGB}$ para resaltar los pixeles donde se detecta únicamente luz roja.

La matriz de imagen de 4 dimensiones, imagen $(x, y)_{RGB}$ puede dividirse en tres matrices independientes imagen $(x, y)_R$, $(x, y)_G$ y $(x, y)_B$, donde cada una de ellas cuantifica entre 0 y 255 la intensidad de los colores rojo, verde y azul en cada píxel (x, y), respectivamente. Estas tres matrices son apreciables en la Fig. 33.

Para conseguir el efecto de filtrado, para cada pixel de la imagen (x, y) se resta a la matriz de componentes rojos R, indicada como matriz imagen $(x, y)_R$, la media de las componentes azul B imagen $(x, y)_B$ y verde G imagen $(x, y)_G$ para cada píxel; tal y como se explica en la expresión siguiente.

$$imagen\ (x, y)_R - \frac{imagen\ (x, y)_G + imagen\ (x, y)_B}{2} = imagen01\ (x, y) \tag{47}$$

El efecto de la aplicación de este filtro es el de identificar los pixeles con valores de color puramente rojos ($R = 255, G = 0, B = 0$), evitando detectar zonas con colores compuestos basados en el rojo, tales como el magenta ($R = 255, G = 0, B = 255$) ó amarillo ($R = 255, G = 255, B = 0$).

Así pues, aquellos pixeles que presenten valores altos en la matriz imagen $(x, y)_R$ además de en algunas de las matrices imagen $(x, y)_G$ o imagen $(x, y)_B$ no se consideran puramente rojos, y su valor asignado en la matriz filtrada imagen01 se verá reducido.

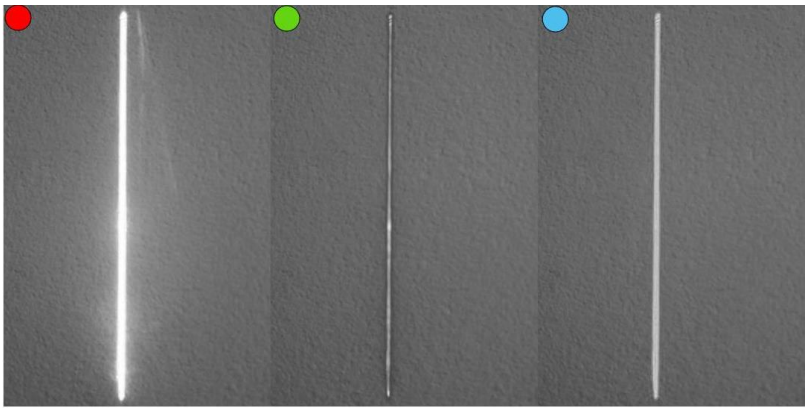


Fig. 33. Ejemplo de imagen separada en RGB de un láser rojo.

2) Resta de imágenes iluminada y no iluminada

El objetivo de esta etapa de procesado es el de reducir el valor de los pixeles no iluminados por el láser en la matriz resultante, idealmente asignando a estos pixeles un valor de 0. Para ello, se hace uso de los datos matriciales correspondientes a la imagen iluminada (MatrizOn), restándoles la matriz de la imagen no iluminada (MatrizOff). Considerando que ambas imágenes han sido capturadas en condiciones de luz ambiental similares, el resultado del proceso de substracción corresponderá idealmente con una matriz donde los píxeles iluminados serán los únicos con valores distintos de cero. En la práctica, y con la finalidad de discriminar pixeles "ruidosos", se considera que únicamente aquellos píxeles con un valor $imagen(x,y) > 150$ han sido iluminados por el láser. A modo de ejemplo, el efecto de los píxeles considerados ruidosos puede apreciarse en la Fig. 34, donde el efecto de reflexión del láser en los bordes del objeto genera zonas parcialmente iluminadas fuera del haz vertical.

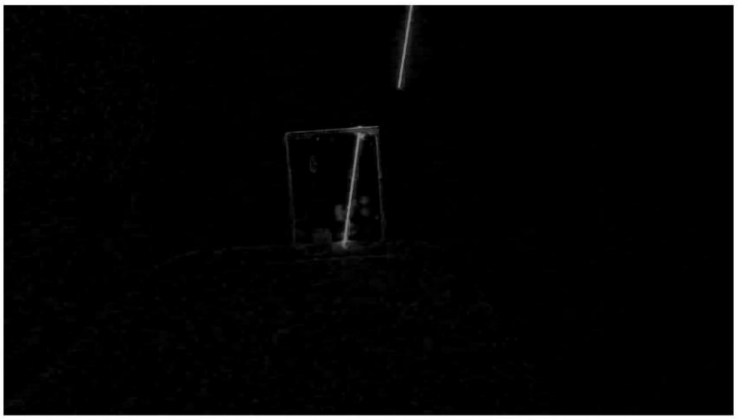


Fig. 34. Imagen diferencia tras pasar ambas imágenes por el filtro RGB.

3) Filtro gaussiano

La aplicación de un filtro gaussiano a la imagen diferencia persigue suavizar la imagen, reduciendo la variación de intensidad lumínica entre pixeles vecinos, eliminando el ruido producido por la reflexión del haz del láser en la superficie del objeto. Este filtro gaussiano se basa en la convolución de una máscara (kernel) ponderada por la función gaussiana y la imagen [11]. La función gaussiana viene dada por la siguiente expresión:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{48}$$

El kernel que se va a utilizar es de tipo 3×3 y se utiliza una sigma de 0,5 para modelar la función tal que el kernel gaussiano viene expresado por la expresión (49).

$$Kernel\ Gaussiano = \begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix} \quad (49)$$

El filtro realizado suaviza los cambios en la imagen como se aprecia en la Fig. 35.

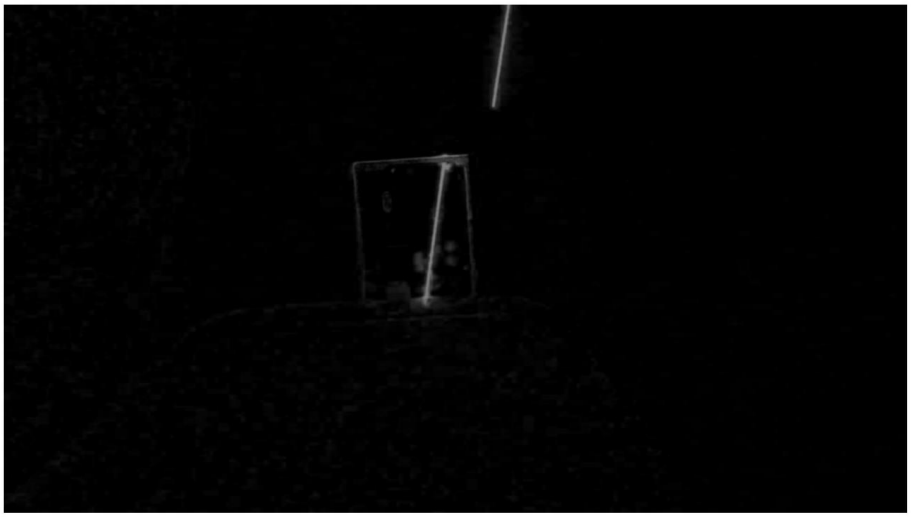


Fig. 35. Imagen resultante del filtrado Gaussiano realizado.

4) <u>Búsqueda de máximos (I)</u>

A partir de esta imagen filtrada, se identifican los valores máximos que puedan existir realizando un barrido por columnas. La identificación de estos máximos permite encontrar los principales puntos de la línea láser, pero también conlleva la identificación de puntos a modo de "ruido" en zonas más oscuras como se ve en la Fig. 36. Resultado de la búsqueda de máximos en la imagen.

Fig. 36. Resultado de la búsqueda de máximos en la imagen.

5) <u>Detección de bordes mediante gradiente</u>

Con el objetivo de identificar trazados continuos de láser con la forma del objeto y descartar falsas discontinuidades se aplica un proceso de detección de bordes mediante el uso de gradientes.

Los bordes en una imagen digital se definen como transiciones entre dos regiones de niveles de grises distintos. Para lograr la detección de bordes en una imagen bidimensional se realiza la primera derivada de la imagen en filas y columnas. Se obtienen así dos derivadas parciales, la derivada parcial en el eje horizontal y la derivada parcial en el eje vertical.

La derivada de una señal continua representa las variaciones locales en dicha señal con respecto a una variable. En este caso, dicha variable es la intensidad de luminosidad y se deriva respecto a la posición que ocupa en la imagen.

Cuando se habla de una función bidimensional $f(x, y)$, su derivada es un vector que apunta en la dirección de la máxima variación de $f(x, y)$ de cuyo módulo es proporcional a esta variación máxima. [12]

Con el fin de obtener la primera derivada parcial en el eje horizontal (columnas) se aplica la siguiente definición de derivada:

$$\frac{\partial \mathrm{f}}{\partial \mathrm{x}} \approx \mathrm{f(x,y)} - \mathrm{f(x-1,y)} \tag{50}$$

Dando como resultado los cambios en el eje horizontal del objeto acorde al desplazamiento en el eje X, como se aprecia en la Fig. 38.

Fig. 37. Representación del gradiente horizontal de la imagen.

Se aprecia que, al tratarse de una línea vertical, las variaciones en el eje X son mínimas y los valores obtenidos son variaciones ínfimas poco apreciables.

Para el caso del gradiente vertical (filas) se aplica la siguiente definición de derivada, estudiando las variaciones en el eje Y.

$$\frac{\partial \mathrm{f}}{\partial \mathrm{y}} \approx \mathrm{f(x,y)} - \mathrm{f(x,y-1)} \tag{51}$$

Los cambios que se aprecian en el eje vertical son mucho más pronunciados como se aprecia en la Fig. 38:

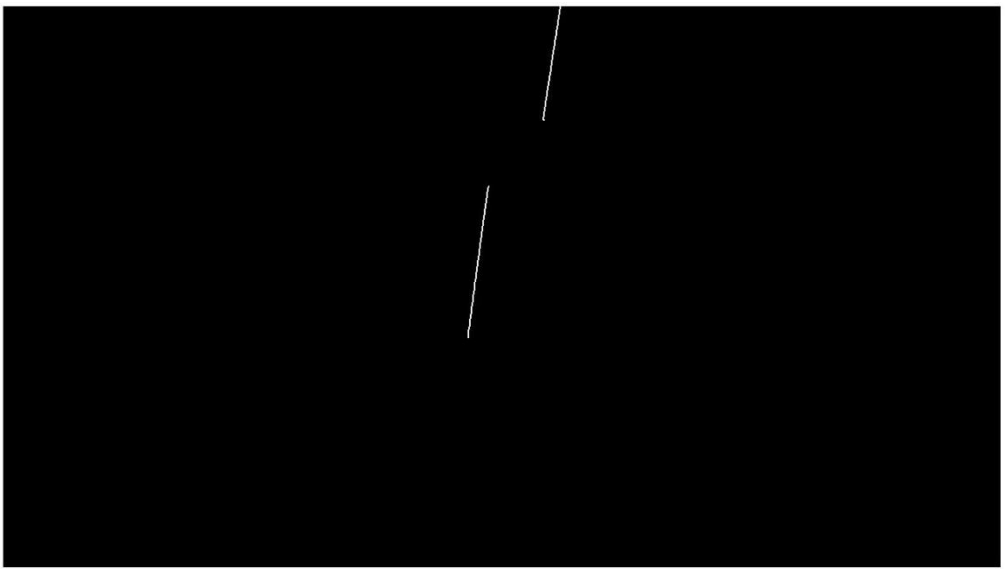


Fig. 38. Representación del gradiente vertical de la imagen.

Mediante estas definiciones se logra la discretización de los vectores gradiente en ambos ejes X e Y, lo que posibilita obtener la magnitud del gradiente completo mediante su definición:

$$Mag[\nabla f(x,y)] = \sqrt{\left(\frac{\partial \mathrm{f}}{\partial \mathrm{y}}\right)^2 + \left(\frac{\partial \mathrm{f}}{\partial \mathrm{y}}\right)^2} \tag{52}$$

Se representa así la magnitud del gradiente, el cual es el resultado final de esta búsqueda de bordes, como se aprecia en la Fig. 39.

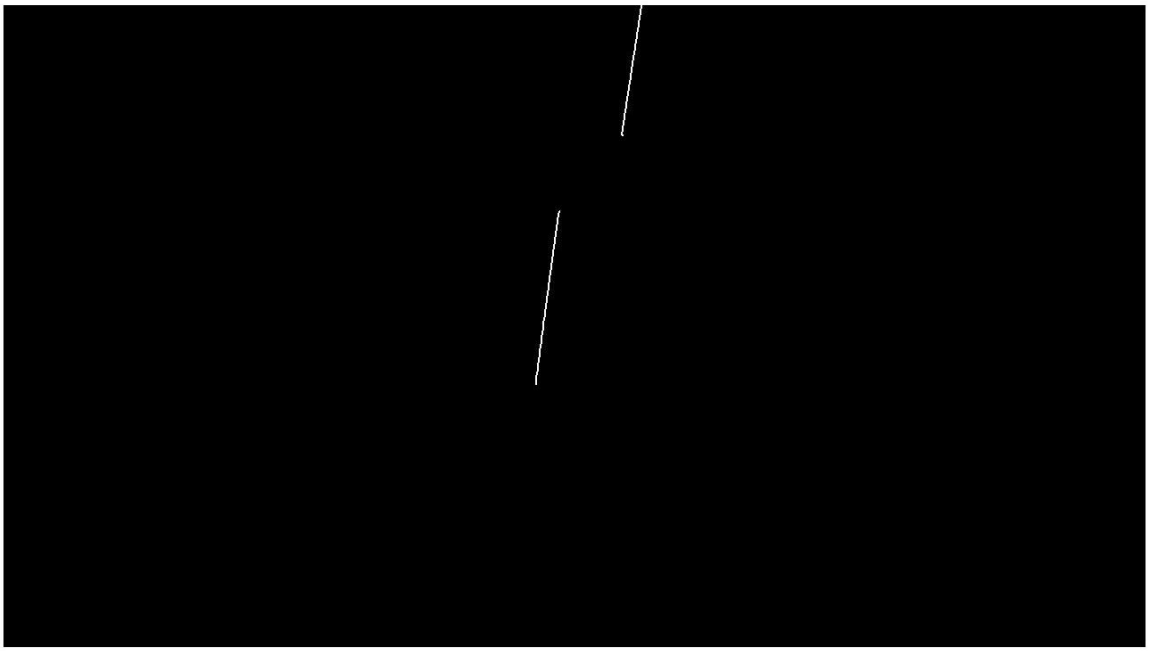


Fig. 39. Representación de la magnitud en el gradiente completo de la imagen.

6) Búsqueda de máximos (II)

A continuación, se realiza una búsqueda de máximos. Si se encuentran varios valores en una misma fila se utiliza la posición intermedia de estos como coordenada; obteniendo un único valor positivo por cada fila, dando como resultado la Fig. 40.

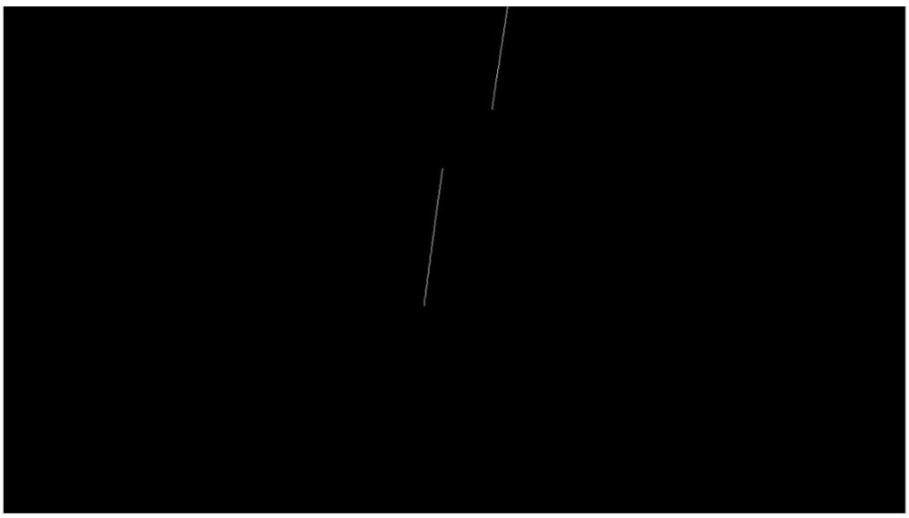


Fig. 40. Resultado de la búsqueda de máximos realizada tras la detección de bordes.

7) Indexación

Se obtiene una imagen binaria como resultado final del filtro, pudiendo interpretarse cada valor positivo como una coordenada del vector. Dicho resultado se ve representado en la Fig. 41.

Con el objetivo de relativizar las coordenadas recién obtenidas en función del centro óptico de la imagen se deberá de indexar estas coordenadas. Para ello, se ha de recorrer la imagen detectando cada coordenada (valor positivo de la imagen) a una matriz de puntos, ajustando los valores a coordenadas de la cámara teniendo en cuenta la distancia con el centro óptico de los ejes. Este ajuste conlleva la diferencia entre la posición absoluta en pixeles de una coordenada y la posición de los ejes, llegando así a unas coordenadas relativas con las que el algoritmo trabaja.

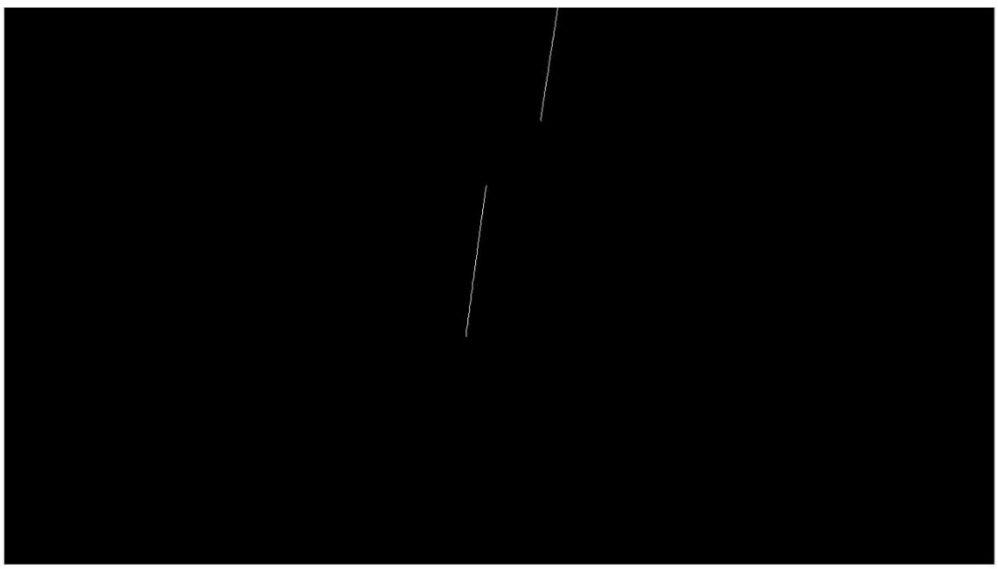


Fig. 41. Línea de puntos resultado final del filtrado.

2.2.15 Generación de coordenadas 3D

De forma previa a la aplicación de las relaciones trigonométricas descritas en el apartado 2.2.4, es necesario realizar una serie de medidas para caracterizar la ubicación espacial de los elementos fijos del sistema, tales como el láser, la cámara y la pared de referencia. La disposición de estos elementos no cambiará durante el proceso de escaneo, por lo que sus relaciones espaciales pueden considerarse valores constantes. Así pues, la medida manual de las posiciones de dichos elementos nos permite conocer, tal y como se indica en la Fig. 42, las siguientes variables iniciales:

- $\boldsymbol{b}$: distancia en el eje x entre el láser y la cámara.
- $\boldsymbol{z_0}$: distancia en el eje z entre el plano láser-cámara y la pared de referencia.

Además, debe recordarse que el valor de la distancia focal de la cámara (F) es conocido, y se ha obtenido mediante el proceso previo de calibración de la cámara, descrito en el apartado 2.2.1.

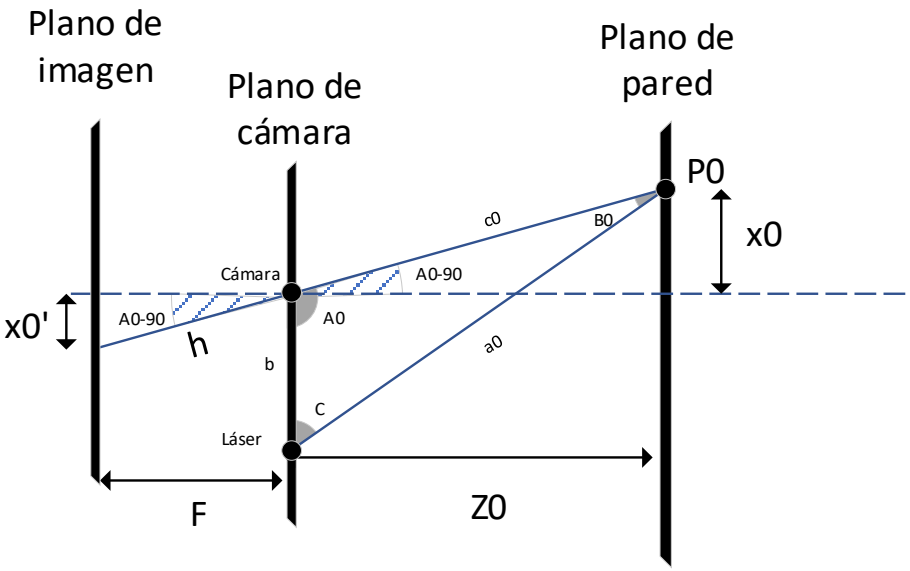


Fig. 42. Vista *XZ* simplificada del sistema de escaneado 3D.

La finalidad del proceso de generación de las coordenadas 3D del objeto no es otro que calcular la distancia del mismo respecto al plano láser-cámara, contenida en la variable z_1. A modo de ejemplo, en los siguientes párrafos se describirá el proceso de obtención de la variable z_1 para un objeto cualquiera, con una separación inicial entre láser y cámara de $b = 180$ nm y una distancia con la pared de referencia de $z_0 = 770$ nm.

Para comenzar, es necesario, como se ha desarrollado en el apartado 2.2.9, caracterizar primero las relaciones trigonométricas entre la tripla láser-pared-cámara, más concretamente las distancias a_0, c_0 y los ángulos A_0, B_0 y C; todas ellas mostradas en la Fig. 42.

En primero de los parámetros calculados es la variable x_0'; correspondiente a la coordenada en la imagen de la posición real x_0 de impacto del láser en la pared de referencia. Es conocido que, por defecto, las imágenes capturadas se almacenan en una matriz de 1280 × 720 × 3, cuyo origen de coordenadas (1, 1, i) corresponde con el píxel de la esquina superior izquierda de la imagen. Para trasladar el origen de coordenadas (0, 0, i) al centro de la imagen, es necesario relativizar las posiciones de los pixeles según la fórmula:

$$x'_0 = x'_0|_{absoluta} - \frac{\text{Resolución horizontal en pixels}}{2} \tag{53}$$

Suponiendo que la imagen capturada original muestra que la coordenada tiene un valor de $x'_0|_{absoluta} = 544$ px, es necesario relativizar su posición utilizando la expresión (53), obteniéndose que la nueva posición es:

$$x'_0 = 544 - 640 = -96 \text{ px} \tag{54}$$

Este valor de la posición x'_0 en el plano de imagen, junto con la distancia focal de $F = 1315$ px obtenida tras el proceso de calibración de la cámara, permite calcular el ángulo A_0 mediante relaciones trigonométricas simples:

$$A_0 = \tan^{-1}\left(\frac{F}{x'_0}\right) = 85.82° \tag{55}$$

Conocidos los valores de F, x'_0 y A_0, la expresión de la cámara *pinhole* permite conocer el valor de distancia x_0 de impacto del láser en la pared; según la fórmula:

$$x_0 = \frac{x'_0 \cdot z_0}{F} = 56.21 \text{ mm} \tag{56}$$

Para continuar con la caracterización geométrica del conjunto láser-pared-cámara, es preciso conocer las distancias c_0 y a_0; las cuales se obtienen de la aplicación del teorema de Pitágoras y la ley de cosenos, respectivamente:

$$c_0 = \sqrt{z_0^2 + x_0^2} = 772 \text{ mm} \tag{57}$$

$$a_0 = \sqrt{b^2 + c_0^2 - 2 \cdot b \cdot c_0 \cdot \cos(A_0)} = 779 \text{ mm} \tag{58}$$

Finalmente, las relaciones entre láser-pared-cámara quedan completamente caracterizadas obteniendo los valores de los ángulos B_0 y C; a través de la aplicación de la ley de senos:

$$\frac{\sin(A_0)}{a_0} = \frac{\sin(B_0)}{b} = \frac{\sin(C)}{c_0} \tag{59}$$

$$B_0 = \sin^{-1}\left[\frac{\sin(A_0)}{a_0} \cdot b\right] = 13.2° \tag{60}$$

$$C = \sin^{-1}\left[\frac{\sin(A_0)}{a_0} \cdot c_0\right] = 81.157° \tag{61}$$

Como puede verse en la Fig. 43, las relaciones entre los tres objetos (láser-pared-cámara) han quedado totalmente caracterizadas, conociéndose así los valores de distancias (a_0, b, c_0) y de ángulos (A_0, B_0, C) entre ellos.

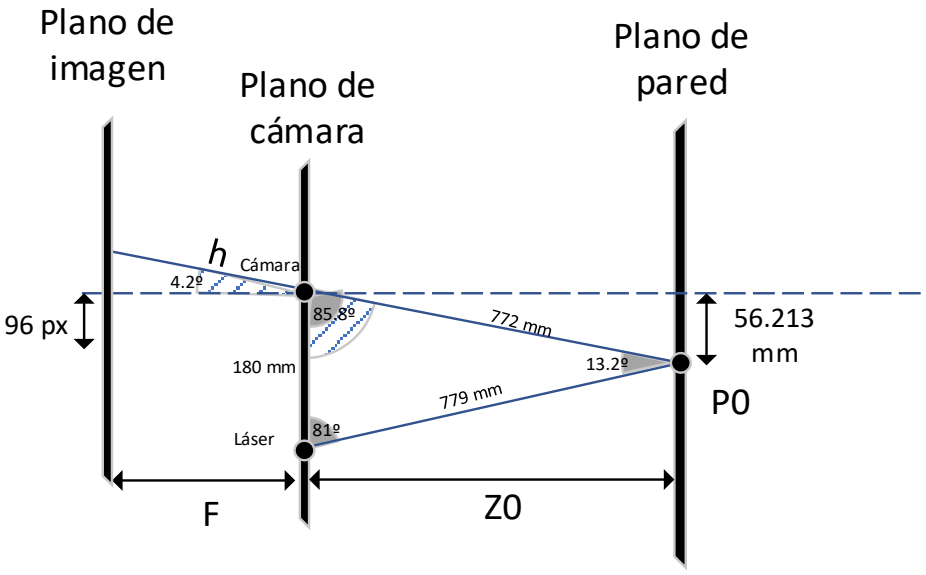


Fig. 43. Vista XZ simplificada del sistema de escaneado 3D, completamente caracterizada para el ejemplo con $b = 180$ mm, $z_0 = 770$ mm y $F = 1315$ px.

Como se ha comentado previamente, el valor de las variables F, b y z_0 se obtiene de forma previa al escaneado y se mantiene constante durante todo el proceso. Ahora, además, es conocido el valor del ángulo C, el cual también se mantendrá invariable durante la ejecución del algoritmo de escaneo 3D del objeto.

Una vez el conjunto de variables F, b y z_0 y C es conocido, es posible proceder con el análisis de las imágenes del objeto tomadas por la cámara. La aplicación del algoritmo mostrado a continuación, permite estimar la distancia z_1, medida desde el origen de coordenadas (0, 0, 0) a la superficie del objeto. El cálculo de esta distancia z_1, como se aprecia en el algoritmo, simplemente necesitará del valor de la variable x'_1, correspondiente a la coordenada en la imagen de la posición real x_1 de impacto del láser en el objeto.

ALGORITMO DE CÁLCULO DE LA DISTANCIA z_1

Paso I: $$A_1 = 90° - tan^{-1}\left(\frac{x'_1}{F}\right) \quad (62)$$

Paso II: $$B_1 = 180° - (A_1 + C) \quad (63)$$

Paso III: $$c_1 = \frac{\sin(C) \cdot b}{\sin(B_1)} \quad (64)$$

Paso IV: $$z_1 = c_1 \cdot \cos(90° - C) \quad (65)$$

Así pues, siguiendo el algoritmo indicado, para cada imagen tomada del objeto, los pixeles iluminados (x'_1, y'_1) tendrán asignado un valor de distancia respecto al origen de coordenadas z_1, que estimará la superficie del objeto en la línea vertical iluminada por el láser.

Por lo tanto, cada vez que se obtiene una captura del objeto, se aplica el algoritmo de cálculo de la distancia z_1, y se obtiene una representación de los datos similar a la incluida en la Fig. 44. Como puede apreciarse en la figura, el eje horizontal representa la coordenada Y de altura del objeto en pixeles; mientras que el eje vertical corresponde con la distancia Z del objeto y pared respecto al origen de coordenadas en milímetros.

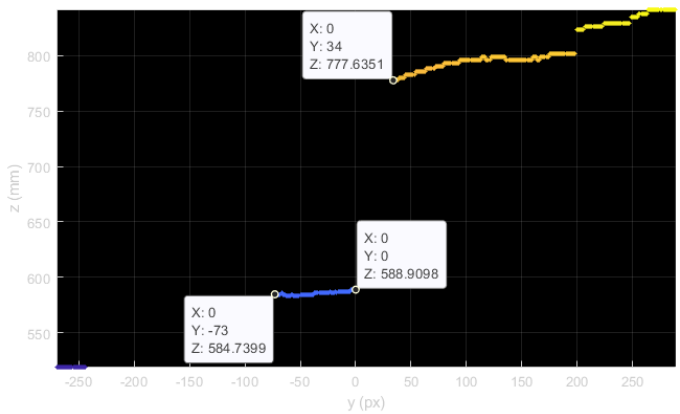


Fig. 44. Resultados del algoritmo de procesado de imagen; eje horizontal Y en píxeles, eje vertical Z en milímetros.

Un análisis rápido de la imagen permite comprobar que el algoritmo de estimación de distancias funciona correctamente. En este caso, la Fig. 44 muestra que le pared referencia se encuentra situada a unos 620 mm de distancia del eje de coordenadas, lo que coincide con el valor conocido de la variable $z_0 = 620$ mm; demostrando un buen nivel de exactitud en el cálculo de las distancias.

Sin embargo, el eje Y, correspondiente a la altura del objeto, sigue presentándose en unidades de píxeles; donde su origen de coordenadas se localiza en la parte inferior de la imagen. Por tanto, para poder construir una matriz de coordenadas con unidades de distancia, es necesario realizar una transformación de los datos de altura, del plano de imagen al plano de objeto.

Esta conversión de unidades se realiza simplemente aplicando un factor de corrección de unidades. Haciendo uso de un objeto de altura conocida, en este caso una caja de alto $h_{obj}(mm) = 27$ mm; junto con las unidades en píxeles de su altura en la imagen $h_{obj}(px) = 73$ px, es aplicable la expresión (66) para obtener el factor de corrección:

$$Factor\ de\ corrección = \frac{h_{obj}(\text{mm})}{h_{obj}(\text{px})} = 0.3698\ \frac{\text{mm}}{\text{px}} \tag{66}$$

Así, el resultado de la aplicación del factor de corrección a la imagen de ejemplo se muestra en la Fig. 45. En esta puede observarse como el objeto a escanear presenta una altura cercana a las 27.7 unidades, lo que se aproxima al valor real de $h_{obj} = 27$ mm. De esta forma, puede asumirse que las unidades del eje Y se corresponden a milímetros, lo que demuestra una buena aplicación del factor de corrección. De forma similar, las dimensiones de altura de la base y la pared han sido corregidas, y sus valores se presentan también en milímetros.

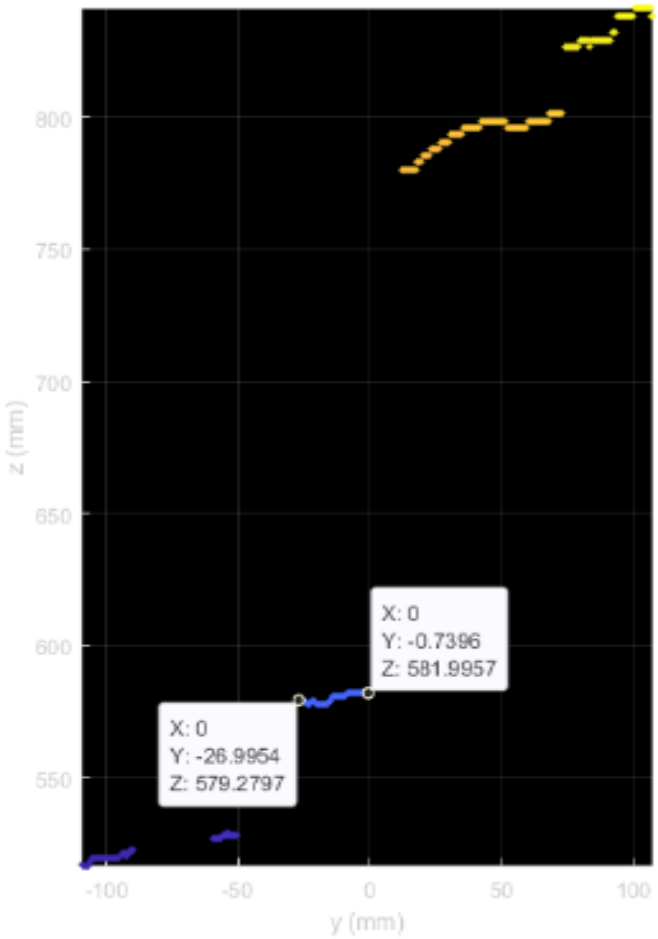


Fig. 45. Resultados de aplicar el factor de corrección; eje horizontal Y en milímetros, eje vertical Z en milímetros.

En este momento del post-procesado de datos, se cuenta con una matriz de datos en la que las coordenadas en Y presentan su origen en el primer píxel iluminado de la imagen; las coordenadas Z tienen el origen en el plano láser-cámara; y las coordenadas en X toman siempre un valor cero. Para facilitar la representación tridimensional de los datos, se desea trasladar el eje de coordenadas (0, 0, 0) al punto central de la plataforma rotatoria. Para llevar a cabo esta operación, mostrada en la Fig. 46, es necesario aplicar la transformación tridimensional indicada en (70).

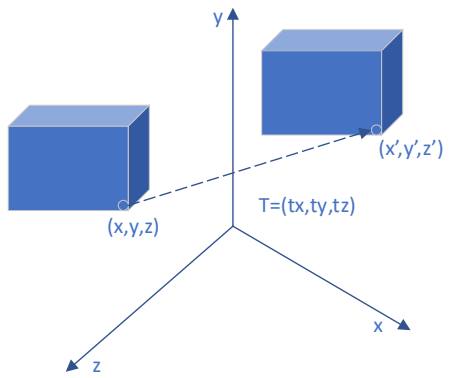


Fig. 46. Imagen conceptual de la translación tridimensional.

La transformación 3D a aplicar puede considerarse de tipo rígida, ya que se conservan los ángulos y longitudes de los elementos a transformar [13]. En este caso, se trata de una traslación, la cual está definida por un vector de tres componentes:

$$T = (t_x, t_y, t_z) \tag{67}$$

Siendo los valores t_x, t_y y t_z las componentes del vector que caracterizan la traslación, de forma que se obtenga como resultado:

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \\ z' &= z + t_z \end{aligned} \tag{68}$$

Donde (x, y, z) son las coordenadas originales de un punto del objeto, y (x', y', z') corresponden con las coordenadas del mismo punto tras realizar la transformación 3D.

Es posible agilizar el cálculo del proceso de traslación operando con los datos en forma matricial, de forma que la operación queda definida por el operador translación M_T, de la forma:

$$M_T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{69}$$

De esta forma, las coordenadas resultantes de cualquier punto Q' pueden calcularse a través de la expresión:

$$Q' = M_T \cdot Q = \begin{bmatrix} x' \\ y' \\ z' \\ 1' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{70}$$

En el caso del ejemplo tratado, el vector de traslación puede definirse como T = (0, 26.99, - 579), desplazando el origen de la imagen al punto central de la plataforma y dando lugar a la matriz de translación (68); tal y como se muestra en la Fig. 46.

Así pues, el valor de las coordenadas (x', y', z') trasladadas al nuevo origen, ubicado en el centro de la plataforma móvil, se obtendrán de la operación matricial siguiente:

$$Q' = M_T \cdot Q = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 26.99 \\ 0 & 0 & 1 & -579 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{71}$$

Por último, se transforman estas coordenadas mediante una rotación sobre el eje Y, ya que las capturas se realizan para cada ángulo de rotación de la plataforma móvil. En este caso, el operador rotación viene expresado por la matriz M_R, la cual presenta la forma indicada en (71).

$$M_R = \begin{bmatrix} \cos(\Theta) & 0 & \sin(\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\Theta) & 0 & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (72)$$

Siendo Θ el ángulo de rotación en cada paso realizado por la plataforma móvil. Este ángulo aumenta con cada paso realizado de forma constante Θ_{step}, por lo que se deberá sumar dicho valor constante en cada iteración.

Aplicando esta transformación a cada punto espacial, se obtiene la coordenada tridimensional característica; y al terminar la rotación completa se almacenan en memoria todos los puntos del objeto. Debe recordarse que en la matriz resultante, las coordenadas de los puntos se presentan de forma relativa al origen de coordenadas situado en el centro de la plataforma móvil.

$$Q'' = M_R \cdot Q' = \begin{bmatrix} x'' \\ y'' \\ z'' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Theta) & 0 & \sin(\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\Theta) & 0 & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \quad (73)$$

Estas transformaciones deben aplicarse tantas veces como el número de capturas adquiridas, lo que se corresponde con el número de desplazamientos de la plataforma móvil. De esta forma, la ejecución del algoritmo de post-procesado de imágenes resultará en una matriz de coordenadas, con dimensiones $720 \times (360°/\theta_{step}) \times 3$.

2.3 Programación

2.3.1 Arduino

El código que se ejecutara en la tarjeta Arduino® es el siguiente:

```
// Descripción de constantes
const int stepPin = 4;        //Definición de pines
const int dirPin = 3;
const int m0Pin = 8;
const int m1Pin = 7;
const int m2Pin = 6;
const int enablePin = 12;
const int sleepPin = 10;
const int resetPin = 11;
const int laserPin = 13;
String entrada= "";                 //Variable de entrada
boolean giro;                       //Sentido del giro
boolean laser;                      //Interruptor laser
//Enviar pulso de Step al controlador respetando los tiempos
//de subida y bajada
void un_paso(){
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(2);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(2);
}
//Encendido del motor respetando los tiempos de reset
// y el tiempo de sleep
void encenderMotor(){
   digitalWrite(enablePin,LOW);
```

```
   digitalWrite(resetPin,HIGH); //RESET
   delay(2); //2ms
   digitalWrite(sleepPin,HIGH); //SLEEP
   delay(2); //2ms
}
//Apagado del motor mediante el pin sleep no necesita respetar tiempos
void apagarMotor(){
   digitalWrite(sleepPin,LOW); //SLEEP
}
//Selecciona el número de posiciones en las que se configura el giro
void pasos(int numero){
  switch(numero){
    case 200:
        digitalWrite(m0Pin,LOW);
        digitalWrite(m1Pin,LOW);
        digitalWrite(m2Pin,LOW);
        break;
    case 400:
        digitalWrite(m0Pin,HIGH);
        digitalWrite(m1Pin,LOW);
        digitalWrite(m2Pin,LOW);
        break;
     case 800:
       digitalWrite(m0Pin,LOW);
       digitalWrite(m1Pin,HIGH);
       digitalWrite(m2Pin,LOW);
       break;
     case 1600:
       digitalWrite(m0Pin,HIGH);
       digitalWrite(m1Pin,HIGH);
       digitalWrite(m2Pin,LOW);
       break;
    case 3200:
      digitalWrite(m0Pin,LOW);
      digitalWrite(m1Pin,LOW);
      digitalWrite(m2Pin,HIGH);
      break;
    case 6400:
      digitalWrite(m0Pin,HIGH);
      digitalWrite(m1Pin,HIGH);
      digitalWrite(m2Pin,HIGH);
      break;
    }
}
//A modo de interruptor invierte el sentido de giro
void cambioGiro(){
  if(giro){
    digitalWrite(dirPin,LOW);
    giro = false;
  }else{
    digitalWrite(dirPin,HIGH);
    giro = true;
  }
}
//A modo de interruptor enciende y apaga el láser
void cambioLaser(){
  if(laser){
    digitalWrite(laserPin,LOW);
    laser = false;
  }else{
    digitalWrite(laserPin,HIGH);
```

```
    laser = true;
  }
}
void setup() {

  //Iniciar conexión puerto serie
  Serial.setTimeout(10);
  Serial.begin(115200);
  Serial.println("ACK");                          //Se envia una señal
a Matlab para dar como iniciada la conexión

  //Declaración de pines de salida
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  pinMode(m0Pin,OUTPUT);
  pinMode(m1Pin,OUTPUT);
  pinMode(m2Pin,OUTPUT);
  pinMode(enablePin,OUTPUT);
  pinMode(resetPin,OUTPUT);
  pinMode(sleepPin,OUTPUT);

  //Configuración inicial
  encenderMotor();
  pasos(200);                                     //Estado inicial 200
pasos
  digitalWrite(dirPin,HIGH);                      //Dirección Inicial
del motor
  giro = true;
  laser = false;
  delayMicroseconds(1);
//Respetar el tiempo al cambiar la dirección
}

void loop() {
      if (Serial.available()){                    //Es ejecutado cada
vez que el puerto serie tiene bytes disponibles
        entrada = Serial.readStringUntil('\n' );  //Lee el puerto
serie hasta que llegue un salto de línea
                                                  //Se utiliza if
debido a que no es posible utilizar switch con strings
        if(entrada=="encenderMotor"){             //Enciende el motor-
> SLEEP y RESET valor alto respetando los tiempos de conmutación
          encenderMotor();
          Serial.println("Motor Encendido");
          delayMicroseconds(2);
        }
        if(entrada=="apagarMotor"){               //Apaga el motor->
SLEEP valor bajo (el motor no consume corriente)
          apagarMotor();
          Serial.println("Motor Apagado");
          delay(20);
        }
        if(entrada=="unPaso"){                    //Dar un paso-> Se
envia un pulso con stepPin al microcontrolador que ejecutara un paso
          un_paso();
          Serial.println("PasoDado");
          delayMicroseconds(2);
        }
        if(entrada=="200pasos"){                  //Configura m0, m1 y
m2 para seleccionar 200 pasos
          pasos(200);
```

```
          Serial.println("200 Pasos");
          delay(20);
        }
        if(entrada=="400pasos"){                    //Configura m0, m1 y
m2 para seleccionar 400 pasos
          pasos(400);
          Serial.println("400 Pasos");
          delay(20);
          }
        if(entrada=="800pasos"){                    //Configura m0, m1 y
m2 para seleccionar 800 pasos
          pasos(800);
          Serial.println("800 Pasos");
          delay(20);
        }
        if(entrada=="1600pasos"){                   //Configura m0, m1 y
m2 para seleccionar 1600 pasos
          pasos(1600);
          Serial.println("1600 Pasos");
          delay(20);
        }
        if(entrada=="3200pasos"){                   //Configura m0, m1 y
m2 para seleccionar 3200 pasos
          pasos(3200);
          Serial.println("3200 Pasos");
          delay(20);
        }
        if(entrada=="6400pasos"){                   //Configura m0, m1 y
m2 para seleccionar 6400 pasos
          pasos(6400);
          Serial.println("6400 Pasos");
          delay(20);
        }
        if(entrada=="cambioGiro"){                  //Cambia el sentido
de giro modificando dirPin
          cambioGiro();
          Serial.println("Sentido de giro cambiado");
          delay(20);
        }
           if(entrada=="cambioLaser"){                  //Cambia el
sentido de giro modificando dirPin
          cambioLaser();
          Serial.println("Estado de laser modificado");
          delay(20);
        }
        entrada="";                                 //Limpiar el string
de entrada
      }
      }
```

2.3.2 Matlab

El código de filtrado utilizado se inspira en la máscara de filtrado utilizada por otras implementaciones habituales de escáneres 3D [14].

En primer lugar, se declaran todos los puertos a utilizar, iniciando la comunicación Arduino® Matlab®.

```
% Limpiar variables anteriores
```

```
clear all

% Limpiar el puerto por si he quedado abierto

delete(instrfind({'Port'},{'COM3'}))

% Crear objeto puerto serie

s = serial('COM3','BaudRate',115200,'Terminator','CR/LF');

% Se define el puerto serie con 115200 baudios (Máximo)

% Además se define el terminador CR/LF = '\n'

% Por lo que todas las escrituras del puerto serie finalizarán

% Con un retorno de carro

% Si existe un error de lectura se imprime en pantalla

warning('off','MATLAB:serial:fscanf:unsuccessfulRead');

% Se inicia la comunicación

fopen(s);

% Se espera a que Arduino este preparado:

% Cuando begin no sea un string vacio el ciclo finaliza

begin='';

while isempty(begin)

    begin=fscanf(s,'%s');       %Actualizar variable con los datos del puerto

end
```

Seguidamente, se declaran todas las constantes del sistema.

```
% Tamaño de imagen

imgSize = [720 1280];

% Distancia focal (px)

f = 1315;
```

```
% Distancia Laser-Camara (mm)

b0 = 195;

% Angulo del laser

C0 = 74.65;

% Factor escalado eje Y

Y_multply = 0.3913;

% Numero de pasos (iteraciones)

N_pasos = 200;

% Angulo que avanza cada paso

Ang_pasos=360/N_pasos;

% Vector desplazamiento 3D (punto central de la plataforma)

x_trans = 0;

y_trans = -84.12;

z_trans = -407.7;

% Puntos maximos por escaneo

MaxPoints = imgSize(1);

% Declaracion Array de coordenadas

cordW = zeros(MaxPoints,3,Main_loops);
```

Se comienza con el ciclo principal del sistema:

```
for Tm = 1:Main_loops
```

El primer proceso del ciclo es llevar a cabo el filtrado laser:

```
    % Imagen frameOn

    % Encender LED
```

```
fwrite(s,'encenderLED');

% Esperar confirmación

while(get(s,'BytesAvailable')==0)

end

a = fscanf(s,'%s');

% Encender Webcam y configurar resolución

cam = webcam(2);

cam.AvailableResolutions;

cam.Resolution='1280x720';

% Tomar captura

frameOn=snapshot(cam);

pause(0.25)

% Cerrar webcam

clear cam;

% Imagen frameOff

% Apagar LED

fwrite(s,'apagarLED');

% Esperar confirmación

while(get(s,'BytesAvailable')==0)

end

a = fscanf(s,'%s');

% Encender Webcam y configurar resolución

cam = webcam(2);

cam.AvailableResolutions;
```

```
cam.Resolution='1280x720';

% Tomar captura

frameOff=snapshot(cam);

pause(0.25)

% Cerrar webcam

clear cam;

% Filtro color RGB

frameongray=frameOn(:,:,1)-((frameOn(:,:,2)+ frameOn(:,:,3))/2);

frameoffgray = frameOff(:,:,1)-((frameOff(:,:,2)+ frameOff(:,:,3))/2);

IM_GRAY=frameongray-frameoffgray;

% Filtro Gaussiano

IM_GAUSS=imgaussfilt(IM_GRAY,1);

% Búsqueda de máximos

for iy = 1:imgSize(1)

xme = max(IM_GAUSS(iy,:));

xme1 = max(find(IM_GAUSS(iy,:)==xme));

IM_MAX(iy,xme1) = 1;

end

% Se retiran los puntos separados (ruido)

IM = IM_MAX;

IM = bwareaopen(IM,20);

% Calculo de gradiente

% Iteración columnas

IM_C = zeros(size(IM));
```

```
    for iy = 1:imgSize(1)-1

        for ix = 1:imgSize(2)

            IM_C(iy,ix,:) =IM(iy+1,ix,:)-IM(iy,ix,:);

        end

    end

    % Iteracion filas

    IM_F =zeros(size(IM));

    for iy = 1:imgSize(1)

        for ix = 1:imgSize(2)-1

            IM_F(iy,ix,:) =IM(iy,ix+1,:)-IM(iy,ix,:);

        end

    end

    % Se calcula el gradiente

    IM_G =zeros(size(IM));

    for iy = 1:imgSize(1)-1

        for ix = 1:imgSize(2)-1

            IM_G(iy,ix,:) = sqrt((IM_C(iy,ix,:)^2)+(IM_F(iy,ix,:)^2));

        end

    end

    % Se vuelve a eliminar el ruido

    IM_G = bwareaopen(IM_G,20);

    % Busqueda de máximos en la imagen

    IM_FILTRO = zeros(size(IM_MAX));

    for y0 = 1:imgSize(1)
```

```
        meanx = round(mean(find(IM_G(y0,:)==1)));
        % Busca el punto intermedio de cada fila
        if isnan(meanx) == 1
            % Si no hay ningún punto, devuelve 0
            IM_FILTRO(y0,:)= 0;
        else
            % Si hay puntos localiza el punto intermedio
            IM_FILTRO(y0,meanx)= 1;
        end
    end
```

Se procede a calcular las posiciones relativas del láser detectado:

```
% Se aplica la indexación mediante la posición del punto central de la imagen
% por lo que se obtienen las posiciones relativas
BW=IM_FILTRO;
    counter = 1;
    for i = 1:imgSize(1)
        for i1 = 1:imgSize(2)
            testpt = BW(i,i1);
            if testpt == 1
                LaserP(1,counter) = 640-i1; %EJE X
                LaserP(2,counter) = 360-i;   %EJE Y
                counter = counter+1;
            end
        end
```

```
    end
```

Se transforman estas posiciones relativas en coordenadas gracias a la trigonometría y al factor de conversión de Y:

```
    sizeDisx = size(LaserP);

    for i = 1:sizeDisx(2)

        % Trigonometría calculo de Z

        delta_px_x1 = LaserP(1,i);

        A1 = 90-atand(delta_px_x1/f);

        B1 = 180 - (A1+C0);

        c1 = (sind(C0)*b0)/sind(B1);

        h1 = c1*cosd(90-C0);

        CamCoordPts(3,i)=h1;

        % Factor escalar Y

        CamCoordPts(2,i)=Y_mltply*LaserP(2,i);

    end

    % X se mantiene a valor 0

    CamCoordPts(1,:) = 0;
```

Se transforman estas coordenadas en el plano YZ a coordenadas reales tridimensionales mediante las matrices de transformación de traslación y rotación:

```
    % Se declara la matriz de traslación tridimensional

    Mext = [ 1 0 0 x_trans;

             0 1 0 y_trans;

             0 0 1 z_trans;

             0 0 0 1;];
```

```
    % Se declara la matriz de rotación que se ve modificada en cada giro

    % por lo que se debe corregir el ángulo

    thetaW=(Tm-1)*Ang_per_loop;

    Trans = [ cosd(thetaW),0,sind(thetaW),0;

              0,1,0,0;

              -sind(thetaW),0,cosd(thetaW),0;

              0,0,0,1];

    % Se declara la matriz de coordenadas y un contador

    CordCT = [];

    % Se aplica la matriz de traslación 3D

    for i = 1:sizeDisx(2)

        CordCT(:,i) =
Mext*[CamCoordPts(1,i);CamCoordPts(2,i);CamCoordPts(3,i);1];

    end

    % Se aplica la matriz de rotación 3D

    for i = 1:sizeDisx(2)

        CordW(:,i) = Trans*[CordCT(1,i);CordCT(2,i);CordCT(3,i);1];

    end

    % Se procede a rellenar las coordenadas vacias (lsa que no han

    % detectado ninguna linea) con valores 0

    transCW = transpose(CordW(1:3,:));

    sRc = size(CordW);

    if sRc(2)==MaxPoints

        cordW(:,:,Tm) =transCW;
```

```
    else

        transCW(sRc(2):MaxPoints,:)=0;

        cordW(:,:,Tm) =transCW;

    end
```

Se realiza un paso preparando al sistema para la siguiente iteración y se cierra el ciclo principal:

```
    fwrite(s,'encenderMotor');

    % Se espera a que arduino tenga datos disponibles

    while(get(s,'BytesAvailable')==0)

    end

    % Se obtiene confirmacion

    a = fscanf(s,'%s');

    % Se realizan tanto saltos como sean necesarios, en una configuración

    % de 200 pasos se podrian realizar 100 realizando saltos de 2 pasos

    for i2=1:1:round(Ang_per_loop/1.8)

        fwrite(s,'unPaso');

        while(get(s,'BytesAvailable')==0)

        end

        a = fscanf(s,'%s');

    end

    % Se apaga el motor para que no se sobrecaliente el sistema

    fwrite(s,'apagarMotor');

    while(get(s,'BytesAvailable')==0)

    end

    a = fscanf(s,'%s');
```

```
end
```

Se convierte el resultado en una matriz de tres columnas (x y z) y se exporta como nube de puntos:

```
xyzPoints = [];

scW = size(cordW);

counter = 1;

for i0 = 1:scW(3)

    for i1 = 1:scW(1)

        xyzPoints(counter,:)=cordW(i1,:,i0);

        counter=counter+1;

    end

end

% Eliminación de muestras fuera de la base de rotación se estima que el

% radio de la base de rotación es 55 mm por lo que se calculan los puntos

% fuera de la base y los convertimos en NaN

RADIO=sqrt(xyzPoints(:,1).^2 + xyzPoints(:,3).^2);

IX0=find(RADIO>=50);

xyzPoints(IX0,[1,2,3])=NaN;

% Se define la matriz de coordenadas como una nube de puntos y se

% representa

ptCloud =pointCloud( xyzPoints);

% Se elimina el ruido (puntos externos no unidos)

ptCloud = pcdenoise(ptCloud);

% Se representa mediante matlab
```

```
pcwrite(ptCloud,'PointCloudOutput','PLYFormat','binary');

figure; pcshow(ptCloud);
```

Por último, se cierran los puertos utilizados:

```
fclose(s);

delete(s);

clear s;
```

3 Conclusiones y resultados

Se realizan varios escaneos de objetos diferentes mediante el sistema final para observar las prestaciones de este. Con el fin de comprobar el comportamiento del sistema ante diferentes objetos de distinta complejidad se procede a analizar cuatro objetos diferentes: un conector pequeño, un vaso de café a modo de cilindro, una pelota de tenis como esfera y una figura compleja.

3.1 Objeto pequeño

Para comenzar, se escanea un conector fijado de forma vertical sobre la plataforma, se configura la opción de menor resolución (200 pasos) debido a que es un objeto de pequeñas dimensiones. Aún con esta resolución el sistema tarda 35 minutos en ejecutarse completamente por lo que los próximos ejemplos continúan en la misma línea de resolución.

En la Fig. 47 se aprecia la forma del conector, al ser un objeto pequeño la resolución utilizada es más que suficiente pudiendo apreciar su forma perfectamente.

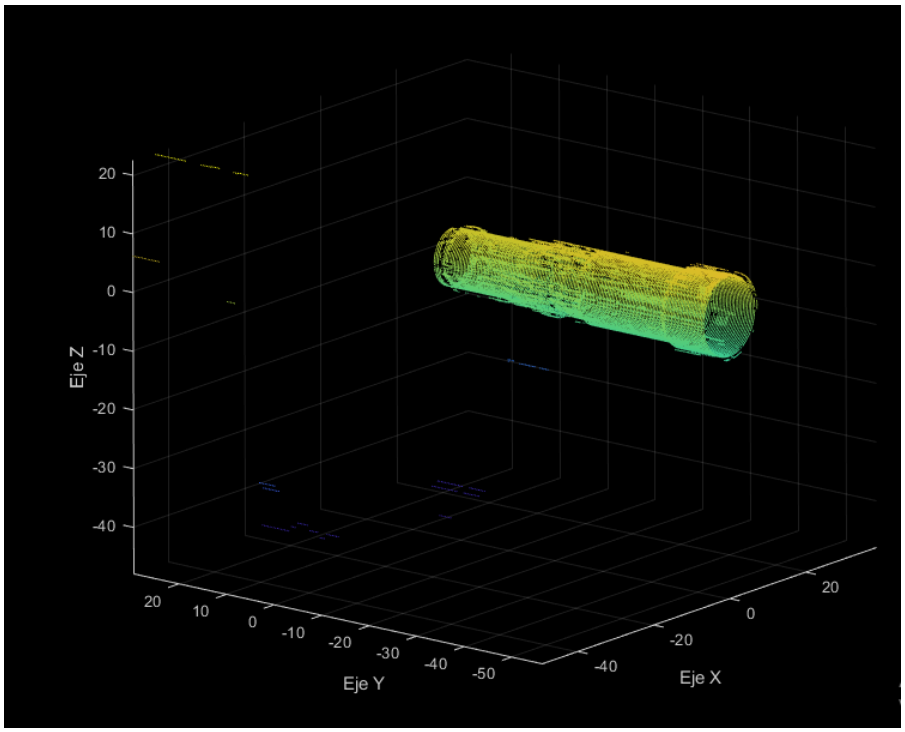


Fig. 47. Escaneo tridimensional del conector

Se comprueban las medidas reales con las sintetizadas, el diámetro del conector se aproxima mucho al real, siendo este 1 cm exacto en la realidad y 1.1 cm en su representación. Obteniendo un margen de error de 1 mm en este ejemplo crítico para el sistema ya que limita con los márgenes de resolución establecidos por el tamaño de pixel del sistema.

Se realiza un segundo escaneo de un objeto pequeño como es una tapa de bolígrafo con resultados muy similares:

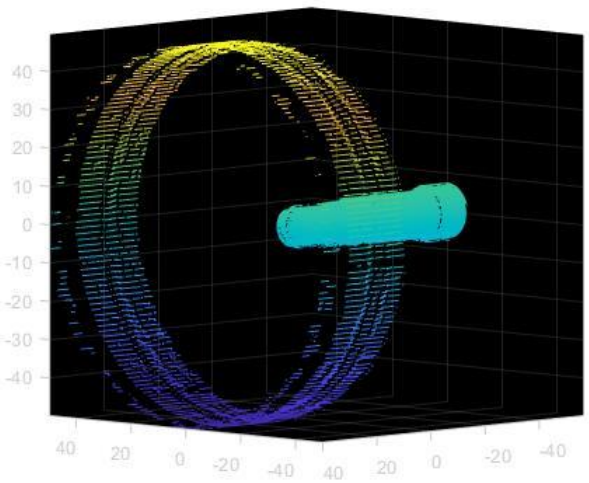
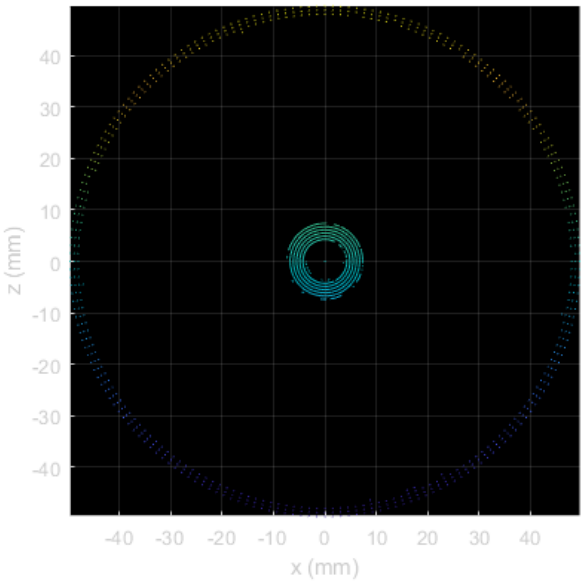
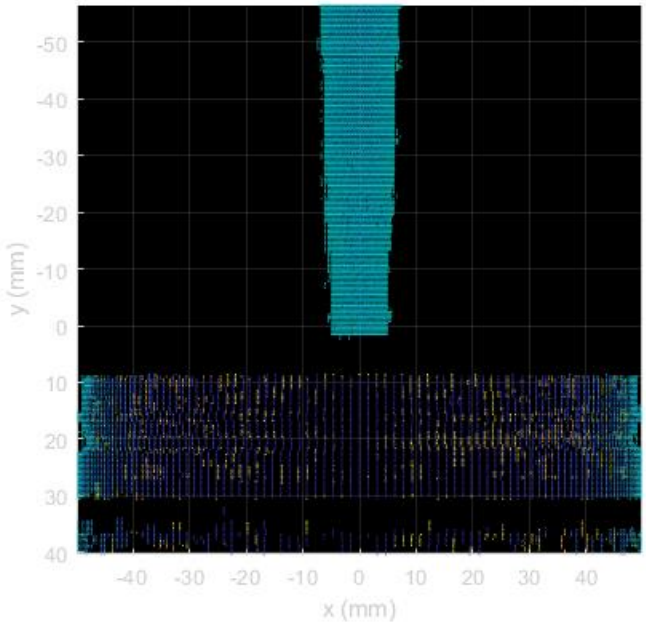


Fig. 48. Escaneo tridimensional de una tapa de bolígrafo

En este ejemplo no se ha realizado un depurado de la imagen y por eso se aprecian las marcas del láser en la pared. Se realizan las vistas de dicho objeto en la Fig. 49.



Vista XZ



Vista XZ

Fig. 49. Vistas XY y XZ obtenidas de la tapa de bolígrafo.

3.2 Cilindro

El vaso de café conlleva la complejidad de ser un objeto grande y se deberían realizar una mayor cantidad de pasos para poder analizar el resultado en profundidad. Por motivos de tiempo se ha realizado un escaneo en 100 pasos para obtener una aproximación rápida se vería con una resolución adecuada como son 800 pasos (lo cual llevaría al sistema un tiempo de ejecución de 2 horas).

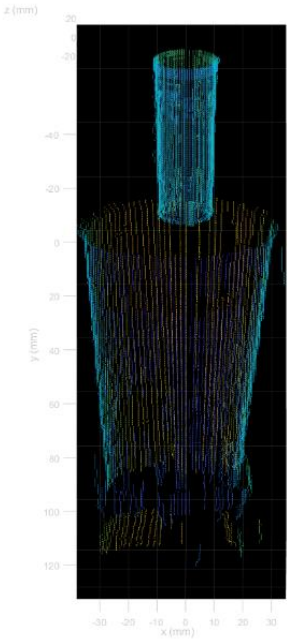


Fig. 50. Escaneo tridimensional de un cilindro (vaso de café)

Como se aprecia en la Fig. 50, no se ha logrado eliminar el efecto del láser en la pared por lo que se aprecia una figura cilíndrica en la parte superior del vaso. Este efecto se debe a las grandes dimensiones del vaso en comparación con los otros objetos analizados que han servido de referencia. Para reparar este error se debe separar aún más las distancias entre la pared y el objeto, por lo que se deben de actualizar las referencias de distancia y calibrar de nuevo el sistema.

Se observa la forma del objeto con total claridad, aunque en comparación con otros resultados se aprecia la escasez de resolución en la nube de puntos, se debe aumentar la resolución de esta para una imagen más detallada del mismo. Este análisis podría servir para medir las dimensiones del vaso, pero no es una nube de puntos detallada y debería de procesarse por suavizados agresivos de superficie para poder representarlo de una forma fiel.

3.3Esfera

Para poder analizar la pelota se dispone una pequeña base circular alrededor de esta, con el fin de obtener una estabilidad constante alrededor de ella. Surge el problema de que dicha esfera está compuesta por un material reflectante que complica el análisis generando mucho más ruido de dispersión que el resto de los objetos dando como resultado el escaneo menos preciso de los realizados. Este ruido se origina en el filtro de luz láser, este debe localizar una única posición de laser por cada columna de imagen, al encontrarse frente a varias posiciones muy dispersas genera posiciones distintas en cada ejecución.

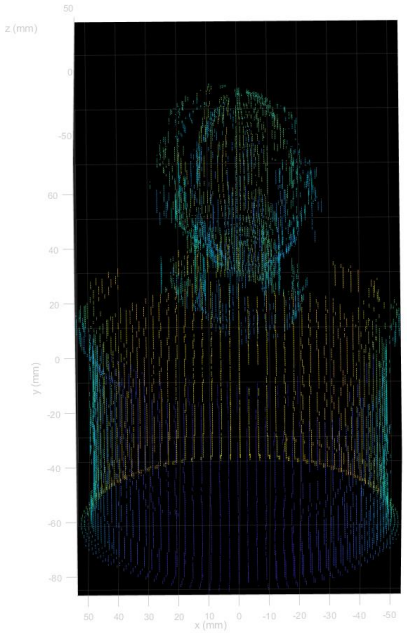


Fig. 51. Escaneo tridimensional resultado de analizar una pelota de tenis.

En la Fig. 51 se aprecia la deformación de la esfera, como se obtiene una forma más ovalada que la real. Parece que la pérdida y dispersión de puntos en la esfera podría deberse a su composición rugosa y su color fosforescente. Aún con esta deformación, se puede apreciar el diámetro de 7 cm de la esfera en su representación cenital XZ en la Fig. 52.

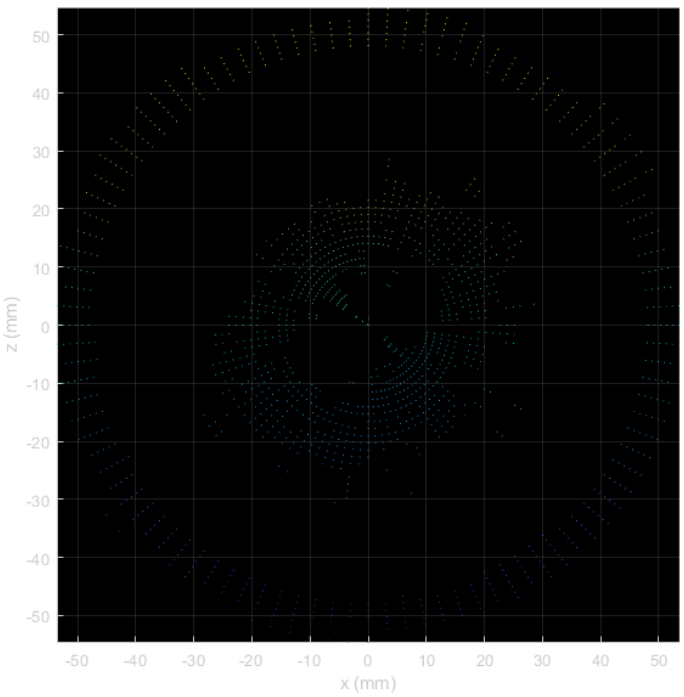


Fig. 52. Vista XZ obtenidas de la pelota de tenis.

Debido al bajo rendimiento del algoritmo de detección del láser observado en el escaneo de la pelota de tenis, se procede a analizar otra esfera como es una pieza de fruta.

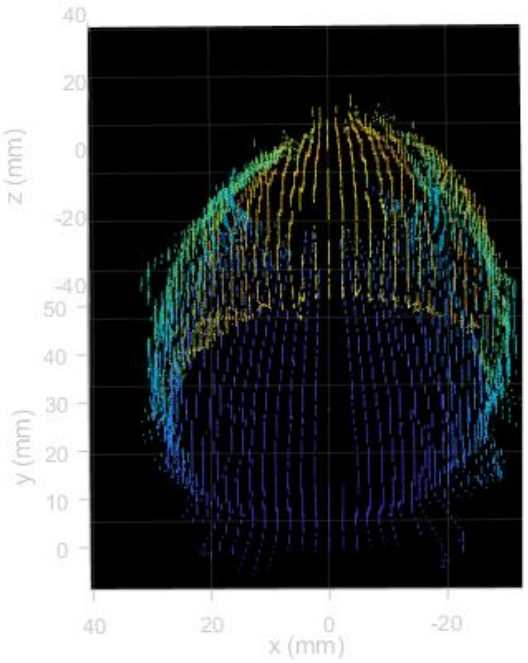


Fig. 53. Escaneo tridimensional de una pieza de fruta.

Se observa un mejor análisis en la pieza de fruta en la Fig. 53, al no ser una esfera perfecta como la pelota, el sistema define mejor sus formas. Aún así, la zona superior del objeto no es representada con fidelidad ya que el análisis del láser se corta antes de llegar al final del objeto. Se observa en la vista cenital XZ, el tamaño de la pieza de fruta que es de unos 8 cm de largo y 6 cm de ancho.

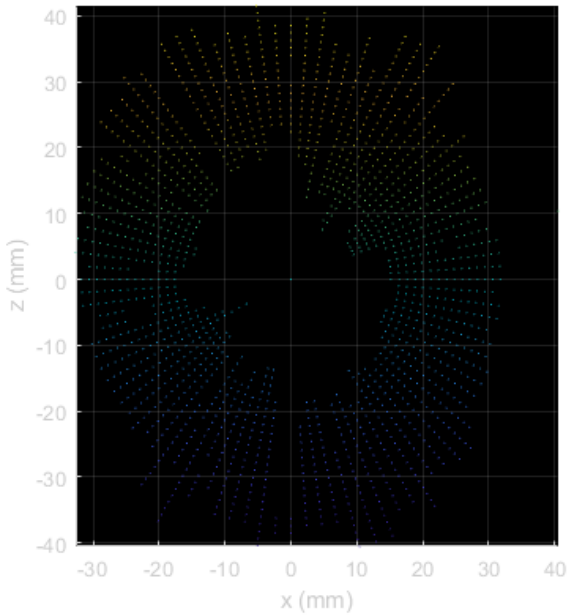


Fig. 54. Vista XZ obtenida de la pieza de fruta.

3.4 Objeto Complejo

Como último objeto, se analiza una figura más compleja como es una figura de una columna jónica en miniatura. Debido a ciertos impedimentos, la resolución de esta se desarrollará en 50 pasos, resolución insuficiente para un objeto de tales dimensiones.

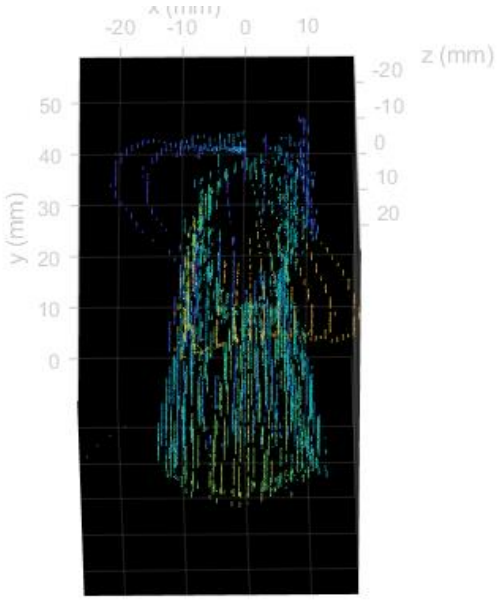


Fig. 55. Escaneo tridimensional de una columna jónica en miniatura.

En la Fig. 55 se llega a apreciar la forma superior de la columna, pero la baja resolución complica su visionado. Se añaden en la Fig. 56 la vista cenital XZ del objeto que coincide con la forma real del objeto.

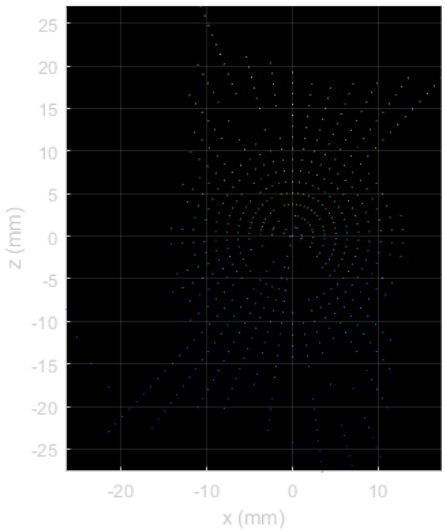


Fig. 56. Vista cenital XZ de la columna jónica en miniatura.

4 Presupuesto

Se realiza una estimación del presupuesto total que se debe gastar en este proyecto con el objetivo de obtener un sistema funcional y completo.

Componente	Precio	N.º	Descripción
Arduino Uno®	19,01 €	1	ARDUINO UNO; SMD R3 ATMEGA328;MCU 8 BITS; ATMEGA328
Motor Bipolar de 200 pasos	15,33 €	1	STEPPER MOTOR HYBRID BIPOLAR 12V; VCC = 12 V; ICC = 330 MA; Θ = 1.8 º
Controlador de motor bipolar	11,82 €	2	MEGAPI STEPPER MOTOR DRIVER V1 CONTROLADOR DE MOTOR PASO A PASO, DRV8825; VCC = 8.2 V ~ 45 V, IOUT(MAX) = 2.5 A; VLOG = 5 V; 16 PINES RESOLUCIÓN=[1/2, 1/4, 1/8, 1/16, 1/32] POR PASO
Láser	5,41 €	1	ADAFRUIT INDUSTRIES LLC LASER DIODE 650NM 5MW 10MM DIA; λ = 650 NM; VCC = 2.8 V ~ 5.2 V; ICC = 25 MA; P = 5 MW

Estructura de giro		1	Realizado mediante impresión 3D
C270 HD WEBCAM	19.50 €	1	C270 HD WEBCAM LOGITECH

Tabla 8. Presupuesto

5 Líneas futuras

El objetivo principal del proyecto se ha podido realizar a pesar de los inconvenientes ocurridos durante el año. Aun así, el proyecto tenía en un principio objetivos mayores como son la impresión de prototipado y una interfaz gráfica.

La estructura de giro esta completada y funciona perfectamente ,pero se debería implementar toda la circuitería de forma permanente ya que actualmente solo es un prototipo. Por lo que una línea de desarrollo consiste en desarrollar todo el sistema de cableado junto al controlador en una placa PCB con conectores para la placa Arduino®.

Para lograr una mejor calidad de impresión debería construirse una estructura cerrada donde introducir el objeto a escanear. El objetivo de esta estructura es el de aislar al sistema de factores ambientales externos, aportando una iluminación óptima para la implementación del algoritmo de detección de la línea.

Uno de los mayores inconvenientes del sistema es su largo tiempo de espera entre captura y captura debido al sistema de procesado de imagen, para optimizar los tiempos se debería instar una depuración del código de Matlab®. Se podría implementar un sistema pipeline con varios hilos de ejecución, separando etapas como la captura de imagen, el filtrado láser o las matrices de

transformación en ejecuciones paralelas. Con este método se optimizaría el tiempo entre capturas y, por lo tanto, el tiempo total de un escaneo.

Por último, un hilo de desarrollo podría centrarse en otros métodos de escaneo como pueden ser el uso de luz estructurada mediante patrones, los cuales permiten un escaneo más veloz, pero con menor precisión. Sería interesante observar la comparativa entre ambos escaneos.

6 Relación de Acrónimos

GUI	Interfaz gráfica de usuario
IDE	Entorno de desarrollo integrado
HD	Alta definición
USB	Bus universal de datos
PCB	Placa de circuito impreso
UART	Puerto universal asíncrono receptor transmisor
TTL	Lógica Transistor a Transistor
CR/LF	Retorno de carro + Salto de línea

7 Bibliografía

[1] C. Teutsch, Model-based Analysis and Evaluation of Point Sets from Optical 3D Laser Scanners, 2007.

[2] Arduino Corporation, «https://www.arduino.cc/,» [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction#. [Último acceso: 1 Junio 2020].

[3] Mathworks, «Mathworks,» Septiembre 2020. [En línea]. Available: https://es.mathworks.com/products/computer-vision.html. [Último acceso: 2 Septiembre 2020].

[4] Mathworks, «Mathworks,» MathWorks Image Acquisition Toolbox Team, Septiembre 2020. [En línea]. Available: https://es.mathworks.com/matlabcentral/fileexchange/49824-matlab-support-package-for-ip-cameras. [Último acceso: 2 Septiembre 2020].

[5] Mathworks, «Mathworks,» Mathworks, Septiembre 2020. [En línea]. Available: https://es.mathworks.com/products/image-acquisition.html. [Último acceso: 2 Septiembre 2020].

[6] Texas Instrument, «Ti.com,» Julio 2014. [En línea]. Available: https://www.ti.com/lit/ds/symlink/drv8825.pdf. [Último acceso: 2020 9 2].

[7] Adafruit Industries, «Digikey,» 10 5 2015. [En línea]. Available: https://www.digikey.es/es/datasheets/adafruit-industries-llc/adafruit-industries-llc-1054_web. [Último acceso: 2 09 2020].

[8] A. Kircher, Ars Magna Lucis et Umbrae, Roma, 1645.

[9] MatterHackers, «matterhacker.com,» [En línea]. Available: https://www.matterhackers.com/store/l/peopoly-moai-130-laser-sla-3d-printer-kit/sk/MNP9N877. [Último acceso: 2 Octubre 2020].

[10] F. Conti, Motores paso a paso, Alsina, 2013.

[11] R. W. R.C. González, Digital Image Processing, Pearson Prentice Hall, 2008.

[12] J. M. P. a. Y. Lu, Wiley Encyclopedia of Computer Science and Engineering, John Wiley & Sons, Inc., 2007.

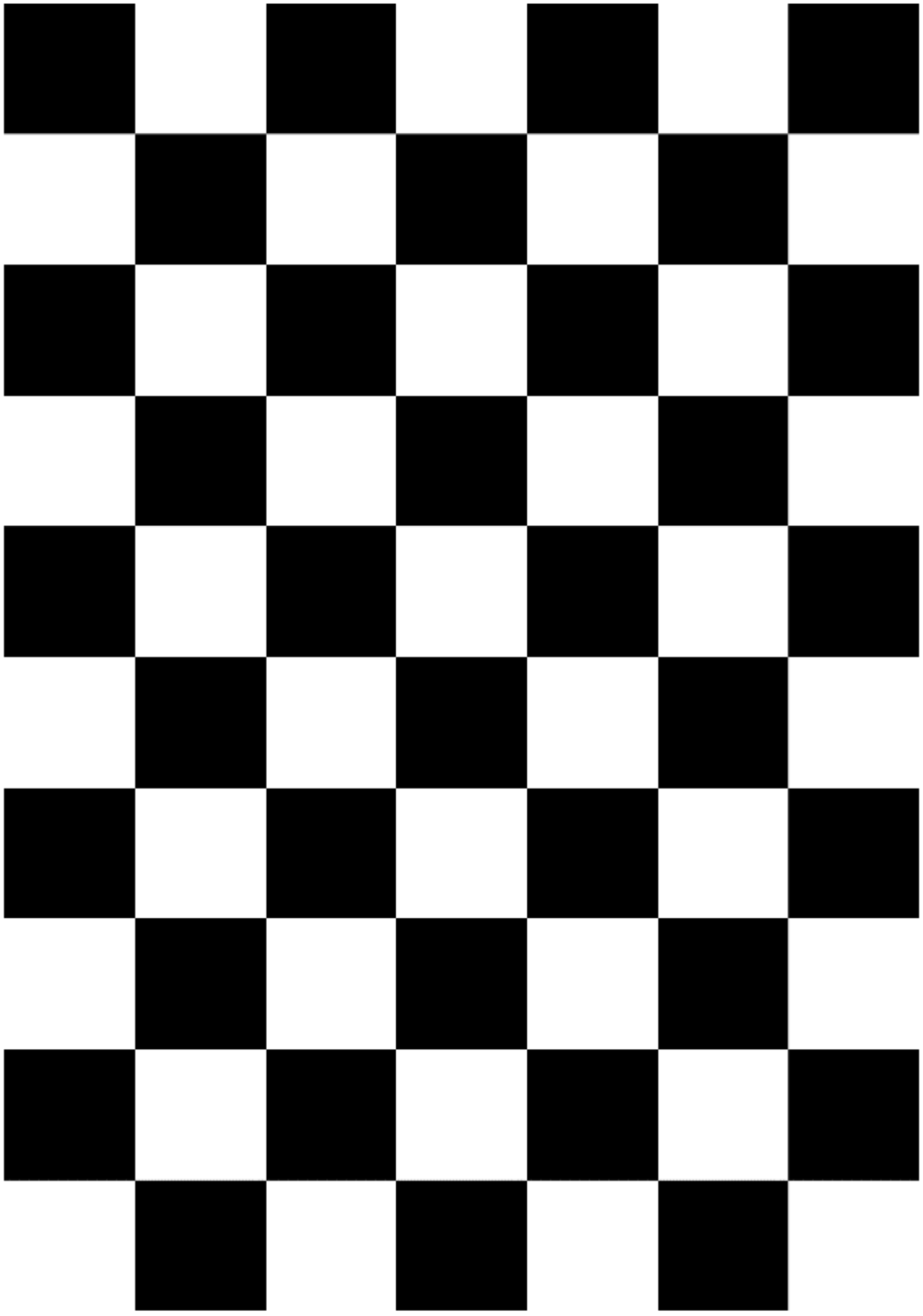
[13] G. Hartman, Fundaments of Matrix Algebra, APEX Calculus, 2011.

[14] J. Lomba, «Research Gate,» [En línea]. Available: https://www.researchgate.net/publication/321668948_Rapid_3D_Printing_for_your_Lab_How_to_make_a_DIY_low-cost_3D_Scanner. [Último acceso: 10 Octubre 2020].

[15] Arduino Foundation, «www.arduino.cc,» 2020. [En línea]. Available: https://www.arduino.cc/en/Guide/Introduction#.

Anexo 1

Cuadrícula de calibración de 2,5 cm de lado



Anexo 2

Hoja de características del microcontrolador DRV8825 de Texas Instruments

TEXAS INSTRUMENTS
DRV8825
SLVSA73F – APRIL 2010 – REVISED JULY 2014

DRV8825 Stepper Motor Controller IC

1 Features

- PWM Microstepping Stepper Motor Driver
  - Built-In Microstepping Indexer
  - Up to 1/32 Microstepping
- Multiple Decay Modes
  - Mixed Decay
  - Slow Decay
  - Fast Decay
- 8.2-V to 45-V Operating Supply Voltage Range
- 2.5-A Maximum Drive Current at 24 V and T_A = 25°C
- Simple STEP/DIR Interface
- Low Current Sleep Mode
- Built-In 3.3-V Reference Output
- Small Package and Footprint
- Protection Features
  - Overcurrent Protection (OCP)
  - Thermal Shutdown (TSD)
  - VM Undervoltage Lockout (UVLO)
  - Fault Condition Indication Pin (nFAULT)

2 Applications

- Automatic Teller Machines
- Money Handling Machines
- Video Security Cameras
- Printers
- Scanners
- Office Automation Machines
- Gaming Machines
- Factory Automation
- Robotics

3 Description

The DRV8825 provides an integrated motor driver solution for printers, scanners, and other automated equipment applications. The device has two H-bridge drivers and a microstepping indexer, and is intended to drive a bipolar stepper motor. The output driver block consists of N-channel power MOSFET's configured as full H-bridges to drive the motor windings. The DRV8825 is capable of driving up to 2.5 A of current from each output (with proper heat sinking, at 24 V and 25°C).

A simple STEP/DIR interface allows easy interfacing to controller circuits. Mode pins allow for configuration of the motor in full-step up to 1/32-step modes. Decay mode is configurable so that slow decay, fast decay, or mixed decay can be used. A low-power sleep mode is provided which shuts down internal circuitry to achieve very low quiescent current draw. This sleep mode can be set using a dedicated nSLEEP pin.

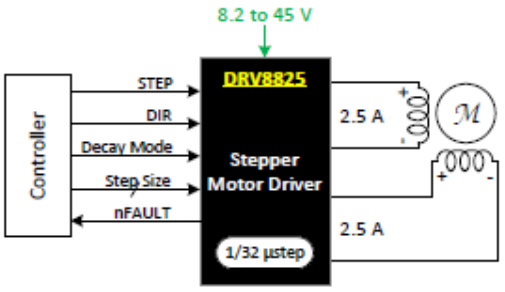
Internal shutdown functions are provided for overcurrent, short circuit, under voltage lockout and over temperature. Fault conditions are indicated via the nFAULT pin.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
DRV8825	HTSSOP (28)	9.70 mm × 6.40 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

4 Simplified Schematic



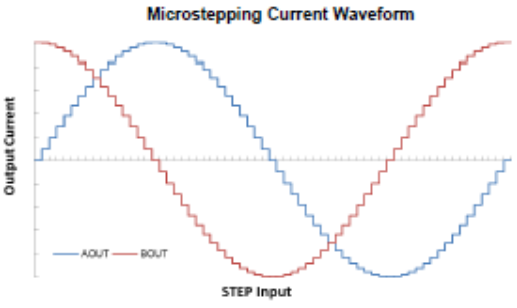
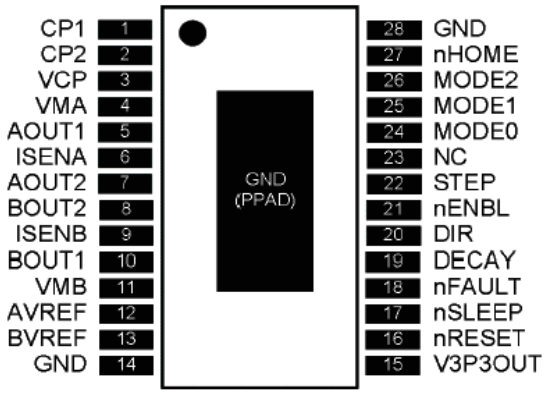


Table of Contents

5 Revision History

Changes from Revision E (August 2013) to Revision F	Page
• Added new sections and reordered data sheet to fit new TI flow	1
• Updated pin descriptions	3
• Added power supply ramp rate and updated ISENSEx pin voltage in *Absolute Maximum Ratings*	4
• Updated V_{IL} voltage minimum and typical in *Electrical Characteristics*	6
• Updated I_{IN} and t_{DEG} in *Electrical Characteristics*	6

6 Pin Configuration and Functions



Pin Functions

PIN		I/O(1)	DESCRIPTION	EXTERNAL COMPONENTS OR CONNECTIONS
NAME	NO.			
POWER AND GROUND				
CP1	1	I/O	Charge pump flying capacitor	Connect a 0.01-µF 50-V capacitor between CP1 and CP2.
CP2	2	I/O	Charge pump flying capacitor	
GND	14, 28	—	Device ground	
VCP	3	I/O	High-side gate drive voltage	Connect a 0.1-µF 16-V ceramic capacitor and a 1-MΩ resistor to VM.
VMA	4	—	Bridge A power supply	Connect to motor supply (8.2 to 45 V). Both pins must be connected to the same supply, bypassed with a 0.1-µF capacitor to GND, and connected to appropriate bulk capacitance.
VMB	11	—	Bridge B power supply	
V3P3OUT	15	O	3.3-V regulator output	Bypass to GND with a 0.47-µF 6.3-V ceramic capacitor. Can be used to supply VREF.
CONTROL				
AVREF	12	I	Bridge A current set reference input	Reference voltage for winding current set. Normally AVREF and BVREF are connected to the same voltage. Can be connected to V3P3OUT.
BVREF	13	I	Bridge B current set reference input	
DECAY	19	I	Decay mode	Low = slow decay, open = mixed decay, high = fast decay. Internal pulldown and pullup.
DIR	20	I	Direction input	Level sets the direction of stepping. Internal pulldown.
MODE0	24	I	Microstep mode 0	MODE0 through MODE2 set the step mode - full, 1/2, 1/4, 1/8/ 1/16, or 1/32 step. Internal pulldown.
MODE1	25	I	Microstep mode 1	
MODE2	26	I	Microstep mode 2	
NC	23	—	No connect	Leave this pin unconnected.
nENBL	21	I	Enable input	Logic high to disable device outputs and indexer operation, logic low to enable. Internal pulldown.
nRESET	16	I	Reset input	Active-low reset input initializes the indexer logic and disables the H-bridge outputs. Internal pulldown.
nSLEEP	17	I	Sleep mode input	Logic high to enable device, logic low to enter low-power sleep mode. Internal pulldown.
STEP	22	I	Step input	Rising edge causes the indexer to move one step. Internal pulldown.
STATUS				
nFAULT	18	OD	Fault	Logic low when in fault condition (overtemp, overcurrent)

(1) Directions: I = input, O = output, OD = open-drain output, IO = input/output

Pin Functions (continued)

PIN		I/O[1]	DESCRIPTION	EXTERNAL COMPONENTS OR CONNECTIONS
NAME	NO.			
nHOME	27	OD	Home position	Logic low when at home state of step table
OUTPUT				
AOUT1	5	O	Bridge A output 1	Connect to bipolar stepper motor winding A. Positive current is AOUT1 → AOUT2
AOUT2	7	O	Bridge A output 2	
BOUT1	10	O	Bridge B output 1	Connect to bipolar stepper motor winding B. Positive current is BOUT1 → BOUT2
BOUT2	8	O	Bridge B output 2	
ISENA	6	I/O	Bridge A ground / Isense	Connect to current sense resistor for bridge A.
ISENB	9	I/O	Bridge B ground / Isense	Connect to current sense resistor for bridge B.

7 Specifications

7.1 Absolute Maximum Ratings[1][2]

		MIN	MAX	UNIT
$V_{(VMx)}$	Power supply voltage	–0.3	47	V
	Power supply ramp rate		1	V/µs
	Digital pin voltage	–0.5	7	V
$V_{(xVREF)}$	Input voltage	–0.3	4	V
	ISENSEx pin voltage[3]	–0.8	0.8	V
	Peak motor drive output current, t < 1 µs		Internally limited	A
	Continuous motor drive output current[4]	0	2.5	A
	Continuous total power dissipation	See *Thermal Information*		
T_J	Operating junction temperature range	–40	150	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions* is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to network ground terminal.
(3) Transients of ±1 V for less than 25 ns are acceptable
(4) Power dissipation and thermal limits must be observed.

7.2 Handling Ratings

			MIN	MAX	UNIT
T_{stg}	Storage temperature range		–60	150	°C
$V_{(ESD)}$	Electrostatic discharge	Human body model (HBM), per ANSI/ESDA/JEDEC JS-001, all pins[1]	–2000	2000	V
		Charged device model (CDM), per JEDEC specification JESD22-C101, all pins[2]	–500	500	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

7.3 Recommended Operating Conditions

		MIN	NOM	MAX	UNIT
$V_{(VMx)}$	Motor power supply voltage range[1]	8.2		45	V
$V_{(VREF)}$	VREF input voltage[2]	1		3.5	V
I_{V3P3}	V3P3OUT load current	0		1	mA

(1) All V_M pins must be connected to the same supply voltage.
(2) Operational at VREF between 0 to 1 V, but accuracy is degraded.

7.4 Thermal Information

THERMAL METRIC[1]		DRV8825 PWP 28 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance[2]	31.6	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance[3]	15.9	
$R_{\theta JB}$	Junction-to-board thermal resistance[4]	5.6	
ψ_{JT}	Junction-to-top characterization parameter[5]	0.2	
ψ_{JB}	Junction-to-board characterization parameter[6]	5.5	
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance[7]	1.4	

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.
(2) The junction-to-ambient thermal resistance under natural convection is obtained in a simulation on a JEDEC-standard, high-K board, as specified in JESD51-7, in an environment described in JESD51-2a.
(3) The junction-to-case (top) thermal resistance is obtained by simulating a cold plate test on the package top. No specific JEDEC-standard test exists, but a close description can be found in the ANSI SEMI standard G30-88.
(4) The junction-to-board thermal resistance is obtained by simulating in an environment with a ring cold plate fixture to control the PCB temperature, as described in JESD51-8.
(5) The junction-to-top characterization parameter, ψ_{JT}, estimates the junction temperature of a device in a real system and is extracted from the simulation data for obtaining θ_{JA}, using a procedure described in JESD51-2a (sections 6 and 7).
(6) The junction-to-board characterization parameter, ψ_{JB}, estimates the junction temperature of a device in a real system and is extracted from the simulation data for obtaining θ_{JA}, using a procedure described in JESD51-2a (sections 6 and 7).
(7) The junction-to-case (bottom) thermal resistance is obtained by simulating a cold plate test on the exposed (power) pad. No specific JEDEC standard test exists, but a close description can be found in the ANSI SEMI standard G30-88.

7.5 Electrical Characteristics

over operating free-air temperature range of –40°C to 85°C (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
POWER SUPPLIES						
I_{VM}	VM operating supply current	$V_{(VMx)}$ = 24 V		5	8	mA
I_{VMQ}	VM sleep mode supply current	$V_{(VMx)}$ = 24 V		10	20	µA
V3P3OUT REGULATOR						
V_{3P3}	V3P3OUT voltage	IOUT = 0 to 1 mA	3.2	3.3	3.4	V
LOGIC-LEVEL INPUTS						
V_{IL}	Input low voltage		0		0.7	V
V_{IH}	Input high voltage		2.2		5.25	V
V_{HYS}	Input hysteresis		0.3	0.45	0.6	V
I_{IL}	Input low current	VIN = 0	–20		20	µA
I_{IH}	Input high current	VIN = 3.3 V			100	µA
R_{PD}	Internal pulldown resistance			100		kΩ
nHOME, nFAULT OUTPUTS (OPEN-DRAIN OUTPUTS)						
V_{OL}	Output low voltage	I_O = 5 mA			0.5	V
I_{OH}	Output high leakage current	V_O = 3.3 V			1	µA
DECAY INPUT						
V_{IL}	Input low threshold voltage	For slow decay mode			0.8	V
V_{IH}	Input high threshold voltage	For fast decay mode	2			V
I_{IN}	Input current		–40		40	µA
R_{PU}	Internal pullup resistance (to 3.3 V)			130		kΩ
R_{PD}	Internal pulldown resistance			80		kΩ
H-BRIDGE FETS						
$R_{DS(ON)}$	HS FET on resistance	$V_{(VMx)}$ = 24 V, I_O = 1 A, T_J = 25°C		0.2		Ω
		$V_{(VMx)}$ = 24 V, I_O = 1 A, T_J = 85°C		0.25	0.32	
	LS FET on resistance	$V_{(VMx)}$ = 24 V, I_O = 1 A, T_J = 25°C		0.2		
		$V_{(VMx)}$ = 24 V, I_O = 1 A, T_J = 85°C		0.25	0.32	
I_{OFF}	Off-state leakage current		–20		20	µA
MOTOR DRIVER						
f_{PWM}	Internal current control PWM frequency			30		kHz
t_{BLANK}	Current sense blanking time			4		µs
t_R	Rise time		30		200	ns
t_F	Fall time		30		200	ns
PROTECTION CIRCUITS						
V_{UVLO}	VM undervoltage lockout voltage	$V_{(VMx)}$ rising		7.8	8.2	V
I_{OCP}	Overcurrent protection trip level		3			A
t_{DEG}	Overcurrent deglitch time			3		µs
t_{TSD}	Thermal shutdown temperature	Die temperature	150	160	180	°C
CURRENT CONTROL						
I_{REF}	xVREF input current	$V_{(xVREF)}$ = 3.3 V	–3		3	µA
V_{TRIP}	xISENSE trip voltage	$V_{(xVREF)}$ = 3.3 V, 100% current setting	635	660	685	mV
ΔI_{TRIP}	Current trip accuracy (relative to programmed value)	$V_{(xVREF)}$ = 3.3 V, 5% current setting	–25%		25%	
		$V_{(xVREF)}$ = 3.3 V, 10% to 34% current setting	–15%		15%	
		$V_{(xVREF)}$ = 3.3 V, 38% to 67% current setting	–10%		10%	
		$V_{(xVREF)}$ = 3.3 V, 71% to 100% current setting	–5%		5%	
A_{ISENSE}	Current sense amplifier gain	Reference only		5		V/V

7.6 Timing Requirements

			MIN	MAX	UNIT
1	f_{STEP}	Step frequency		250	kHz
2	$t_{WH(STEP)}$	Pulse duration, STEP high	1.9		µs
3	$t_{WL(STEP)}$	Pulse duration, STEP low	1.9		µs
4	$t_{SU(STEP)}$	Setup time, command before STEP rising	650		ns
5	$t_{H(STEP)}$	Hold time, command after STEP rising	650		ns
6	t_{ENBL}	Enable time, nENBL active to STEP	650		ns
7	t_{WAKE}	Wakeup time, nSLEEP inactive high to STEP input accepted		1.7	ms

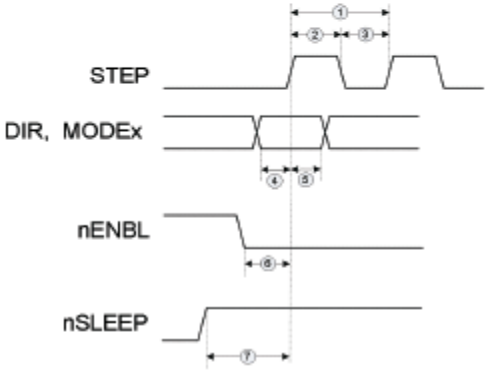


Figure 1. Timing Diagram

7.7 Typical Characteristics

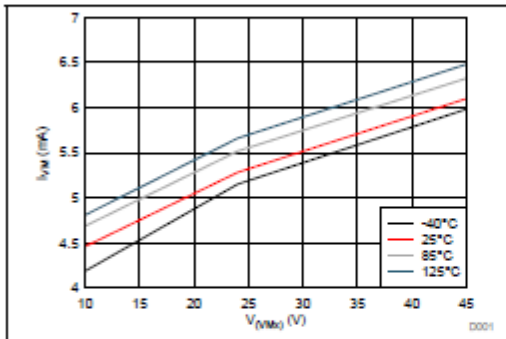


Figure 2. I_{VMx} vs $V_{(VMx)}$

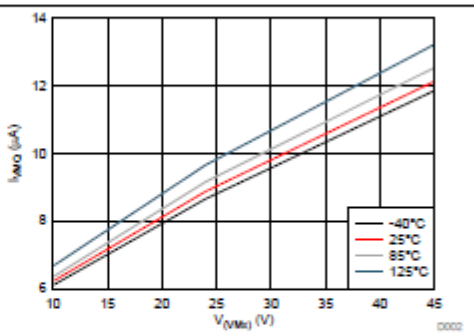


Figure 3. I_{VMxQ} vs $V_{(VMx)}$

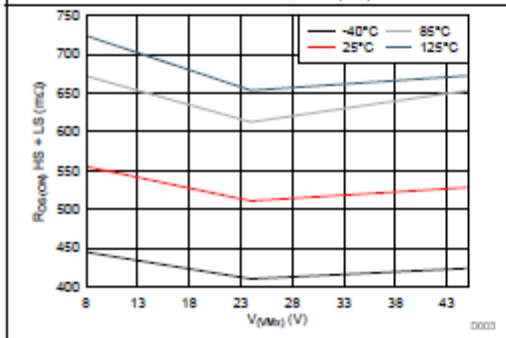


Figure 4. $R_{DS(ON)}$ vs $V_{(VMx)}$

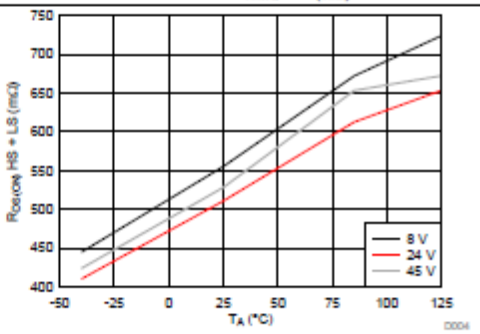


Figure 5. $R_{DS(ON)}$ vs Temperature

8 Detailed Description

8.1 Overview

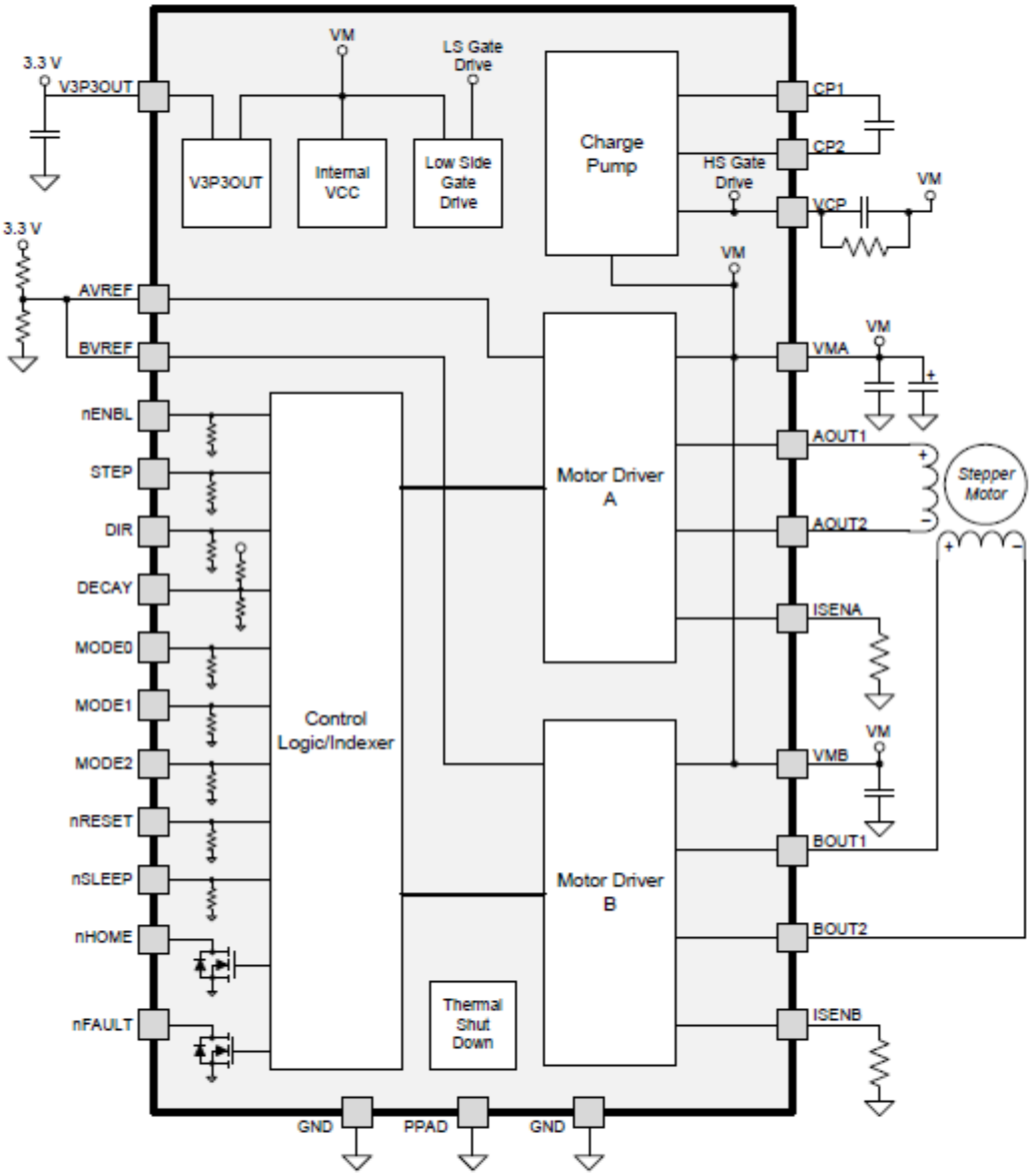
The DRV8825 is an integrated motor driver solution for bipolar stepper motors. The device integrates two NMOS H-bridges, current sense, regulation circuitry, and a microstepping indexer. The DRV8825 can be powered with a supply voltage between 8.2 and 45 V and is capable of providing an output current up to 2.5 A full-scale.

A simple STEP/DIR interface allows for easy interfacing to the controller circuit. The internal indexer is able to execute high-accuracy microstepping without requiring the processor to control the current level.

The current regulation is highly configurable, with three decay modes of operation. Depending on the application requirements, the user can select fast, slow, and mixed decay.

A low-power sleep mode is included which allows the system to save power when not driving the motor.

8.2 Functional Block Diagram



8.3 Feature Description

8.3.1 PWM Motor Drivers

The DRV8825 contains two H-bridge motor drivers with current-control PWM circuitry. Figure 6 shows a block diagram of the motor control circuitry.

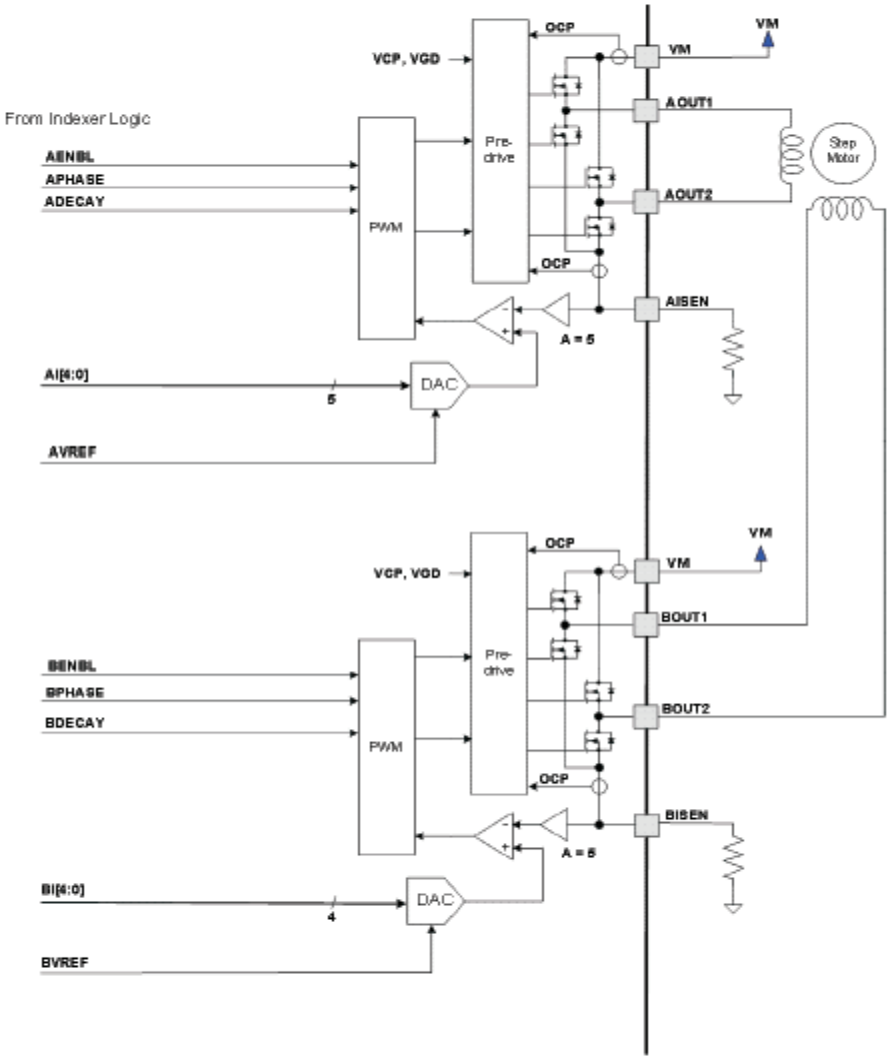


Figure 6. Motor Control Circuitry

Note that there are multiple VM motor power supply pins. All VM pins must be connected together to the motor supply voltage.

Feature Description (continued)

8.3.2 Current Regulation

The current through the motor windings is regulated by a fixed-frequency PWM current regulation, or current chopping. When an H-bridge is enabled, current rises through the winding at a rate dependent on the DC voltage and inductance of the winding. Once the current hits the current chopping threshold, the bridge disables the current until the beginning of the next PWM cycle.

In stepping motors, current regulation is used to vary the current in the two windings in a semi-sinusoidal fashion to provide smooth motion.

The PWM chopping current is set by a comparator which compares the voltage across a current sense resistor connected to the xISEN pins, multiplied by a factor of 5, with a reference voltage. The reference voltage is input from the xVREF pins.

The full-scale (100%) chopping current is calculated in Equation 1.

$$I_{CHOP} = \frac{V_{(xREF)}}{5 \times R_{ISENSE}} \quad (1)$$

Example:

If a 0.25-Ω sense resistor is used and the VREFx pin is 2.5 V, the full-scale (100%) chopping current will be 2.5 V / (5 x 0.25 Ω) = 2 A.

The reference voltage is scaled by an internal DAC that allows fractional stepping of a bipolar stepper motor, as described in the microstepping indexer section below.

8.3.3 Decay Mode

During PWM current chopping, the H-bridge is enabled to drive current through the motor winding until the PWM current chopping threshold is reached. This is shown in Figure 7 as case 1. The current flow direction shown indicates positive current flow.

Once the chopping current threshold is reached, the H-bridge can operate in two different states, fast decay or slow decay.

In fast decay mode, once the PWM chopping current level has been reached, the H-bridge reverses state to allow winding current to flow in a reverse direction. As the winding current approaches 0, the bridge is disabled to prevent any reverse current flow. Fast decay mode is shown in Figure 7 as case 2.

In slow decay mode, winding current is recirculated by enabling both of the low-side FETs in the bridge. This is shown in Figure 7 as case 3.

Feature Description (continued)

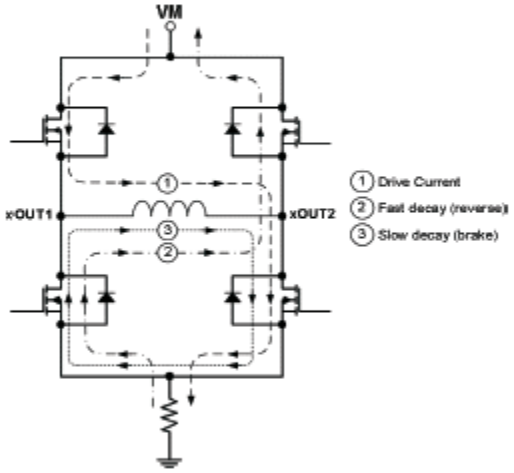


Figure 7. Decay Mode

The DRV8825 supports fast decay, slow decay and a mixed decay mode. Slow, fast, or mixed decay mode is selected by the state of the DECAY pin; logic low selects slow decay, open selects mixed decay operation, and logic high sets fast decay mode. The DECAY pin has both an internal pullup resistor of approximately 130 kΩ and an internal pulldown resistor of approximately 80 kΩ. This sets the mixed decay mode if the pin is left open or undriven.

Mixed decay mode begins as fast decay, but at a fixed period of time (75% of the PWM cycle) switches to slow decay mode for the remainder of the fixed PWM period. This occurs only if the current through the winding is decreasing (per the indexer step table); if the current is increasing, then slow decay is used.

8.3.4 Blanking Time

After the current is enabled in an H-bridge, the voltage on the xISEN pin is ignored for a fixed period of time before enabling the current sense circuitry. This blanking time is fixed at 3.75 µs. Note that the blanking time also sets the minimum on time of the PWM.

8.3.5 Microstepping Indexer

Built-in indexer logic in the DRV8825 allows a number of different stepping configurations. The MODE0 through MODE2 pins are used to configure the stepping format as shown in Table 1.

Table 1. Stepping Format

MODE2	MODE1	MODE0	STEP MODE
0	0	0	Full step (2-phase excitation) with 71% current
0	0	1	1/2 step (1-2 phase excitation)
0	1	0	1/4 step (W1-2 phase excitation)
0	1	1	8 microsteps/step
1	0	0	16 microsteps/step
1	0	1	32 microsteps/step
1	1	0	32 microsteps/step
1	1	1	32 microsteps/step

Table 2 shows the relative current and step directions for different settings of MODEx. At each rising edge of the STEP input, the indexer travels to the next state in the table. The direction is shown with the DIR pin high; if the DIR pin is low the sequence is reversed. Positive current is defined as xOUT1 = positive with respect to xOUT2.

Note that if the step mode is changed while stepping, the indexer will advance to the next valid state for the new MODEx setting at the rising edge of STEP.

The home state is 45°. This state is entered at power-up or application of nRESET. This is shown in Table 2 by the shaded cells. The logic inputs DIR, STEP, nRESET, and MODEx have internal pulldown resistors of 100 kΩ.

Table 2. Relative Current and Step Directions

1/32 STEP	1/16 STEP	1/8 STEP	1/4 STEP	1/2 STEP	FULL STEP 70%	WINDING CURRENT A	WINDING CURRENT B	ELECTRICAL ANGLE
1	1	1	1	1		100%	0%	0
2						100%	5%	3
3	2					100%	10%	6
4						99%	15%	8
5	3	2				98%	20%	11
6						97%	24%	14
7	4					96%	29%	17
8						94%	34%	20
9	5	3	2			92%	38%	23
10						90%	43%	25
11	6					88%	47%	28
12						86%	51%	31
13	7	4				83%	56%	34
14						80%	60%	37
15	8					77%	63%	39
16						74%	67%	42
17	9	5	3	2	1	71%	71%	45
18						67%	74%	48
19	10					63%	77%	51
20						60%	80%	53
21	11	6				56%	83%	56
22						51%	86%	59
23	12					47%	88%	62
24						43%	90%	65
25	13	7	4			38%	92%	68
26						34%	94%	70
27	14					29%	96%	73
28						24%	97%	76
29	15	8				20%	98%	79
30						15%	99%	82
31	16					10%	100%	84
32						5%	100%	87
33	17	9	5	3		0%	100%	90
34						–5%	100%	93
35	18					–10%	100%	96
36						–15%	99%	98
37	19	10				–20%	98%	101
38						–24%	97%	104
39	20					–29%	96%	107

Table 2. Relative Current and Step Directions (continued)

1/32 STEP	1/16 STEP	1/8 STEP	1/4 STEP	1/2 STEP	FULL STEP 70%	WINDING CURRENT A	WINDING CURRENT B	ELECTRICAL ANGLE
40						–34%	94%	110
41	21	11	6			–38%	92%	113
42						–43%	90%	115
43	22					–47%	88%	118
44						–51%	86%	121
45	23	12				–56%	83%	124
46						–60%	80%	127
47	24					–63%	77%	129
48						–67%	74%	132
49	25	13	7	4	2	–71%	71%	135
50						–74%	67%	138
51	26					–77%	63%	141
52						–80%	60%	143
53	27	14				–83%	56%	146
54						–86%	51%	149
55	28					–88%	47%	152
56						–90%	43%	155
57	29	15	8			–92%	38%	158
58						–94%	34%	160
59	30					–96%	29%	163
60						–97%	24%	166
61	31	16				–98%	20%	169
62						–99%	15%	172
63	32					–100%	10%	174
64						–100%	5%	177
65	33	17	9	5		–100%	0%	180
66						–100%	–5%	183
67	34					–100%	–10%	186
68						–99%	–15%	188
69	35	18				–98%	–20%	191
70						–97%	–24%	194
71	36					–96%	–29%	197
72						–94%	–34%	200
73	37	19	10			–92%	–38%	203
74						–90%	–43%	205
75	38					–88%	–47%	208
76						–86%	–51%	211
77	39	20				–83%	–56%	214
78						–80%	–60%	217
79	40					–77%	–63%	219
80						–74%	–67%	222
81	41	21	11	6	3	–71%	–71%	225
82						–67%	–74%	228
83	42					–63%	–77%	231
84						–60%	–80%	233
85	43	22				–56%	–83%	236
86						–51%	–86%	239

Table 2. Relative Current and Step Directions (continued)

1/32 STEP	1/16 STEP	1/8 STEP	1/4 STEP	1/2 STEP	FULL STEP 70%	WINDING CURRENT A	WINDING CURRENT B	ELECTRICAL ANGLE
87	44					–47%	–88%	242
88						–43%	–90%	245
89	45	23	12			–38%	–92%	248
90						–34%	–94%	250
91	46					–29%	–96%	253
92						–24%	–97%	256
93	47	24				–20%	–98%	259
94						–15%	–99%	262
95	48					–10%	–100%	264
96						–5%	–100%	267
97	49	25	13	7		0%	–100%	270
98						5%	–100%	273
99	50					10%	–100%	276
100						15%	–99%	278
101	51	26				20%	–98%	281
102						24%	–97%	284
103	52					29%	–96%	287
104						34%	–94%	290
105	53	27	14			38%	–92%	293
106						43%	–90%	295
107	54					47%	–88%	298
108						51%	–86%	301
109	55	28				56%	–83%	304
110						60%	–80%	307
111	56					63%	–77%	309
112						67%	–74%	312
113	57	29	15	8	4	71%	–71%	315
114						74%	–67%	318
115	58					77%	–63%	321
116						80%	–60%	323
117	59	30				83%	–56%	326
118						86%	–51%	329
119	60					88%	–47%	332
120						90%	–43%	335
121	61	31	16			92%	–38%	338
122						94%	–34%	340
123	62					96%	–29%	343
124						97%	–24%	346
125	63	32				98%	–20%	349
126						99%	–15%	352
127	64					100%	–10%	354
128						100%	–5%	357

8.3.6 nRESET, nENBL, and nSLEEP Operation

The nRESET pin, when driven active low, resets internal logic, and resets the step table to the home position. It also disables the H-bridge drivers. The STEP input is ignored while nRESET is active.

The nENBL pin is used to control the output drivers and enable/disable operation of the indexer. When nENBL is low, the output H-bridges are enabled, and rising edges on the STEP pin are recognized. When nENBL is high, the H-bridges are disabled, the outputs are in a high-impedance state, and the STEP input is ignored.

Driving nSLEEP low will put the device into a low power sleep state. In this state, the H-bridges are disabled, the gate drive charge pump is stopped, the V3P3OUT regulator is disabled, and all internal clocks are stopped. In this state all inputs are ignored until nSLEEP returns inactive high. When returning from sleep mode, some time (approximately 1 ms) needs to pass before applying a STEP input, to allow the internal circuitry to stabilize. Note that nRESET and nENABL have internal pulldown resistors of approximately 100 kΩ. The nSLEEP pin has an internal pulldown resistor of 1 MΩ. nSLEEP and nRESET signals need to be driven to logic high for device operation.

8.3.7 Protection Circuits

The DRV8825 is fully protected against undervoltage, overcurrent, and overtemperature events.

8.3.7.1 Overcurrent Protection (OCP)

An analog current limit circuit on each FET limits the current through the FET by removing the gate drive. If this analog current limit persists for longer than the OCP time, all FETs in the H-bridge will be disabled and the nFAULT pin will be driven low. The device remains disabled until either nRESET pin is applied, or VM is removed and reapplied.

Overcurrent conditions on both high-side and low-side devices; that is, a short to ground, supply, or across the motor winding all result in an overcurrent shutdown. Note that overcurrent protection does not use the current sense circuitry used for PWM current control, and is independent of the I_{SENSE} resistor value or xVREF voltage.

8.3.7.2 Thermal Shutdown (TSD)

If the die temperature exceeds safe limits, all FETs in the H-bridge will be disabled and the nFAULT pin will be driven low. After the die temperature has fallen to a safe level, operation automatically resumes.

8.3.7.3 Undervoltage Lockout (UVLO)

If at any time the voltage on the VM pins falls below the UVLO threshold voltage, all circuitry in the device will be disabled and internal logic will be reset. Operation will resume when $V_{(VMx)}$ rises above the UVLO threshold.

8.4 Device Functional Modes

8.4.1 STEP/DIR Interface

The STEP/DIR interface provides a simple method for advancing through the indexer table. For each rising edge on the STEP pin, the indexer travels to the next state in the table. The direction it moves in the table is determined by the input to the DIR pin. The signals applied to the STEP and DIR pins should not violate the timing diagram specified in Figure 1.

8.4.2 Microstepping

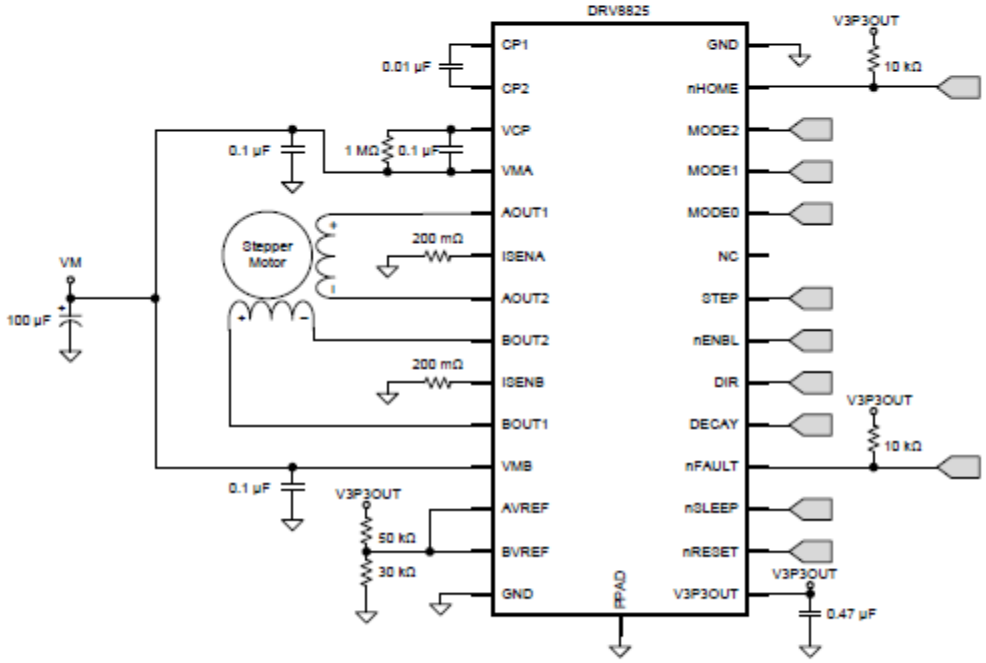
The microstepping indexer allows for a variety of stepping configurations. The state of the indexer is determined by the configuration of the three MODE pins (refer to Table 1 for configuration options). The DRV8825 supports full step up to 1/32 microstepping.

9 Application and Implementation

9.1 Application Information

The DRV8825 is used in bipolar stepper control. The microstepping motor driver provides additional precision and a smooth rotation from the stepper motor. The following design is a common application of the DRV8825.

9.2 Typical Application



9.2.1 Design Requirements

Design Parameter	Reference	Example Value
Supply Voltage	VM	24 V
Motor Winding Resistance	RL	3.9 Ω
Motor Winding Inductance	IL	2.9 mH
Motor Full Step Angle	θstep	1.8°/step
Target Microstepping Level	nm	8 µsteps per step
Target Motor Speed	v	120 rpm
Target Full-Scale Current	IFS	1.25 A

9.2.2 Detailed Design Procedure

9.2.2.1 Stepper Motor Speed

The first step in configuring the DRV8825 requires the desired motor speed and microstepping level. If the target application requires a constant speed, then a square wave with frequency f_{step} must be applied to the STEP pin.

If the target motor startup speed is too high, the motor will not spin. Make sure that the motor can support the target speed or implement an acceleration profile to bring the motor up to speed.

For a desired motor speed (v), microstepping level (n_m), and motor full step angle (θ_{step}),

$$f_{step}\left(\frac{\mu steps}{second}\right) = \frac{v\left(\frac{rotations}{minute}\right) \times 360\left(\frac{^\circ}{rotation}\right) \times n_m\left(\frac{\mu steps}{step}\right)}{60\left(\frac{seconds}{minute}\right) \times \theta_{step}\left(\frac{^\circ}{step}\right)} \quad (2)$$

$$f_{step}\left(\frac{\mu steps}{second}\right) = \frac{120\left(\frac{rotations}{minute}\right) \times 360\left(\frac{^\circ}{rotation}\right) \times 8\left(\frac{\mu steps}{step}\right)}{60\left(\frac{seconds}{minute}\right) \times 1.8\left(\frac{^\circ}{step}\right)} \quad (3)$$

θ_{step} can be found in the stepper motor data sheet or written on the motor itself.

For the DRV8825, the microstepping level is set by the MODE pins and can be any of the settings in Table 1. Higher microstepping will mean a smoother motor motion and less audible noise, but will increase switching losses and require a higher f_{step} to achieve the same motor speed.

9.2.2.2 Current Regulation

In a stepper motor, the set full-scale current (I_{FS}) is the maximum current driven through either winding. This quantity depends on the xVREF analog voltage and the sense resistor value (R_{SENSE}). During stepping, I_{FS} defines the current chopping threshold (I_{TRIP}) for the maximum current step. The gain of DRV8825 is set for 5 V/V.

$$I_{FS}(A) = \frac{xVREF(V)}{A_V \times R_{SENSE}(\Omega)} = \frac{xVREF(V)}{5 \times R_{SENSE}(\Omega)} \quad (4)$$

To achieve I_{FS} = 1.25 A with R_{SENSE} of 0.2 Ω, xVREF should be 1.25 V.

9.2.2.3 Decay Modes

The DRV8825 supports three different decay modes: slow decay, fast decay, and mixed decay. The current through the motor windings is regulated using a fixed-frequency PWM scheme. This means that after any drive phase, when a motor winding current has hit the current chopping threshold (I_{TRIP}), the DRV8825 will place the winding in one of the three decay modes until the PWM cycle has expired. Afterward, a new drive phase starts.

The blanking time, t_{BLANK}, defines the minimum drive time for the current chopping. I_{TRIP} is ignored during t_{BLANK}, so the winding current may overshoot the trip level.

9.2.3 Application Curves

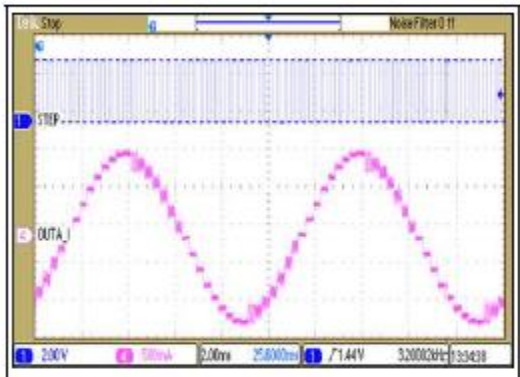


Figure 8. Microstepping Current (Phase A) vs STEP Input, Mixed Decay

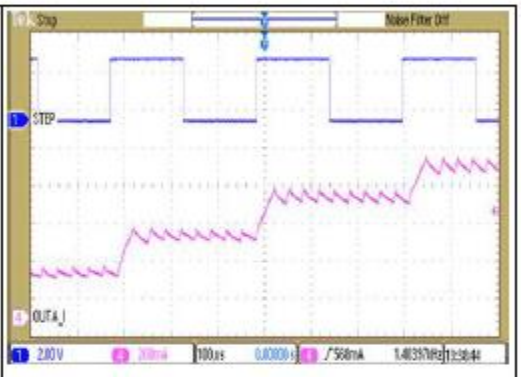


Figure 9. Microstepping Current (Phase A) vs STEP Input, Slow Decay on Increasing Steps

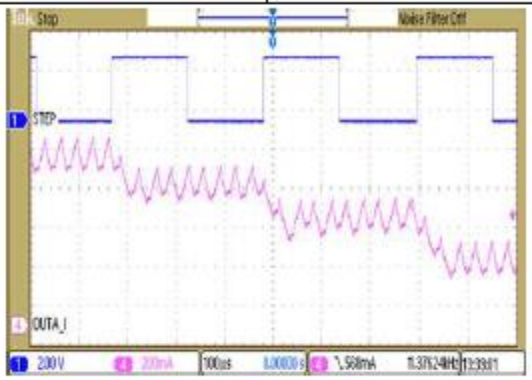


Figure 10. Microstepping Current (Phase A) vs STEP Input, Mixed Decay on Decreasing Steps

10 Power Supply Recommendations

The DRV8825 is designed to operate from an input voltage supply (VMx) range between 8.2 and 45 V. Two 0.1-µF ceramic capacitors rated for VMx must be placed as close as possible to the VMA and VMB pins respectively (one on each pin). In addition to the local decoupling caps, additional bulk capacitance is required and must be sized accordingly to the application requirements.

10.1 Bulk Capacitance

Bulk capacitance sizing is an important factor in motor drive system design. It is dependent on a variety of factors including:

- Type of power supply
- Acceptable supply voltage ripple
- Parasitic inductance in the power supply wiring
- Type of motor (brushed DC, brushless DC, stepper)
- Motor startup current
- Motor braking method

The inductance between the power supply and motor drive system will limit the rate current can change from the power supply. If the local bulk capacitance is too small, the system will respond to excessive current demands or dumps from the motor with a change in voltage. You should size the bulk capacitance to meet acceptable voltage ripple levels.

The data sheet generally provides a recommended value but system level testing is required to determine the appropriate sized bulk capacitor.

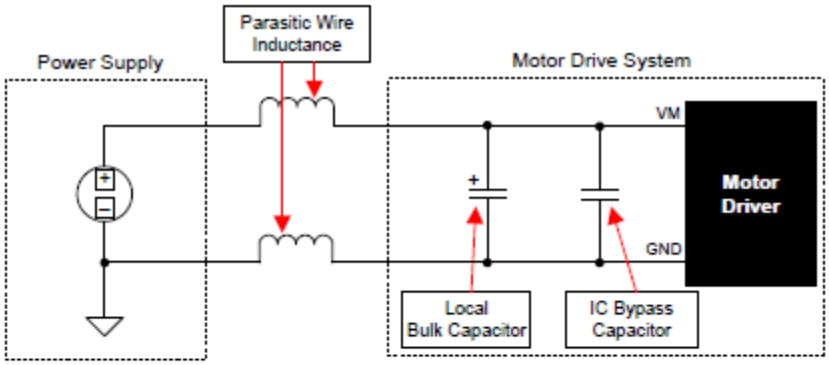


Figure 11. Setup of Motor Drive System With External Power Supply

10.2 Power Supply and Logic Sequencing

There is no specific sequence for powering-up the DRV8825. It is okay for digital input signals to be present before VMx is applied. After VMx is applied to the DRV8825, it begins operation based on the status of the control pins.

11 Layout

11.1 Layout Guidelines

The VMA and VMB pins should be bypassed to GND using low-ESR ceramic bypass capacitors with a recommended value of 0.1-µF rated for VMx. This capacitor should be placed as close to the VMA and VMB pins as possible with a thick trace or ground plane connection to the device GND pin.

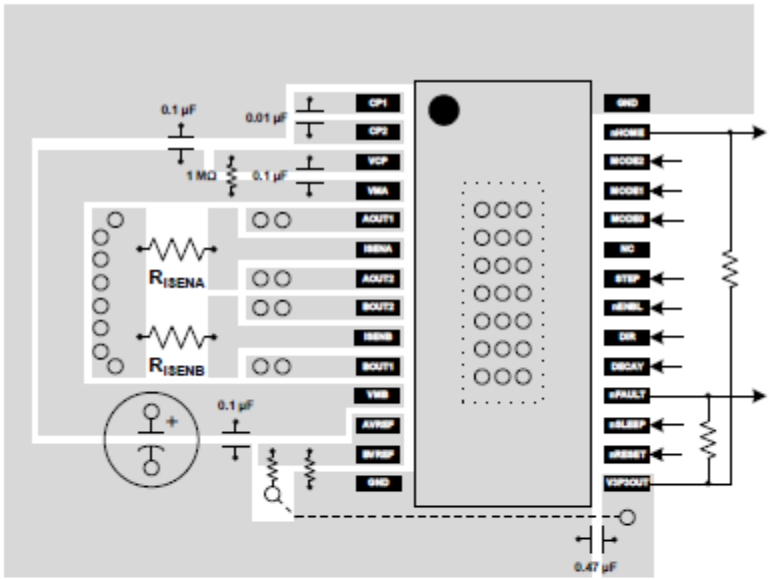
The VMA and VMB pins must be bypassed to ground using an appropriate bulk capacitor. This component may be an electrolytic and should be located close to the DRV8825.

A low-ESR ceramic capacitor must be placed in between the CPL and CPH pins. TI recommends a value of 0.01-µF rated for VMx. Place this component as close to the pins as possible.

A low-ESR ceramic capacitor must be placed in between the VMA and VCP pins. TI recommends a value of 0.1-µF rated for 16 V. Place this component as close to the pins as possible. Also, place a 1-MΩ resistor between VCP and VMA.

Bypass V3P3 to ground with a ceramic capacitor rated 6.3 V. Place this bypass capacitor as close to the pin as possible

11.2 Layout Example



11.3 Thermal Protection

The DRV8825 has thermal shutdown (TSD) as described above. If the die temperature exceeds approximately 150°C, the device will be disabled until the temperature drops to a safe level.

Any tendency of the device to enter TSD is an indication of either excessive power dissipation, insufficient heatsinking, or too high an ambient temperature.

11.3.1 Power Dissipation

Power dissipation in the DRV8825 is dominated by the power dissipated in the output FET resistance, or $R_{DS(ON)}$. Average power dissipation when running a stepper motor can be roughly estimated by Equation 5.

Thermal Protection (continued)

$$P_{TOT} = 4 \times R_{DS(ON)} \times \left(I_{OUT(RMS)}\right)^2 \quad (5)$$

where P_{TOT} is the total power dissipation, $R_{DS(ON)}$ is the resistance of each FET, and $I_{OUT(RMS)}$ is the RMS output current being applied to each winding. $I_{OUT(RMS)}$ is equal to the approximately 0.7x the full-scale output current setting. The factor of 4 comes from the fact that there are two motor windings, and at any instant two FETs are conducting winding current for each winding (one high-side and one low-side).

The maximum amount of power that can be dissipated in the device is dependent on ambient temperature and heatsinking.

Note that $R_{DS(ON)}$ increases with temperature, so as the device heats, the power dissipation increases. This must be taken into consideration when sizing the heatsink.

11.3.2 Heatsinking

The PowerPAD™ package uses an exposed pad to remove heat from the device. For proper operation, this pad must be thermally connected to copper on the PCB to dissipate heat. On a multi-layer PCB with a ground plane, this can be accomplished by adding a number of vias to connect the thermal pad to the ground plane. On PCBs without internal planes, copper area can be added on either side of the PCB to dissipate heat. If the copper area is on the opposite side of the PCB from the device, thermal vias are used to transfer the heat between top and bottom layers.

For details about how to design the PCB, refer to TI application report SLMA002, "PowerPAD™ Thermally Enhanced Package" and TI application brief SLMA004, *PowerPAD™ Made Easy*, available at www.ti.com.

In general, the more copper area that can be provided, the more power can be dissipated. It can be seen that the heatsink effectiveness increases rapidly to about 20 cm^2, then levels off somewhat for larger areas.

12 Device and Documentation Support

12.1 Trademarks

PowerPAD is a trademark of Texas Instruments.

12.2 Electrostatic Discharge Caution

These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

12.3 Glossary

SLYZ022 — *TI Glossary*.

This glossary lists and explains terms, acronyms, and definitions.

13 Mechanical, Packaging, and Orderable Information

The following pages include mechanical, packaging, and orderable information. This information is the most current data available for the designated devices. This data is subject to change without notice and revision of this document. For browser-based versions of this data sheet, refer to the left-hand navigation.

TEXAS INSTRUMENTS

www.ti.com

PACKAGE OPTION ADDENDUM

6-Feb-2020

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
DRV8825PWP	ACTIVE	HTSSOP	PWP	28	50	Green (RoHS & no Sb/Br)	NIPDAU	Level-3-260C-168 HR	-40 to 85	DRV8825	Samples
DRV8825PWPR	ACTIVE	HTSSOP	PWP	28	2000	Green (RoHS & no Sb/Br)	NIPDAU	Level-3-260C-168 HR	-40 to 85	DRV8825	Samples

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) **RoHS:** TI defines "RoHS" to mean semiconductor products that are compliant with the current EU RoHS requirements for all 10 RoHS substances, including the requirement that RoHS substance do not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, "RoHS" products are suitable for use in specified lead-free processes. TI may reference these types of products as "Pb-Free".
RoHS Exempt: TI defines "RoHS Exempt" to mean products that contain lead but are compliant with EU RoHS pursuant to a specific EU RoHS exemption.
Green: TI defines "Green" to mean the content of Chlorine (Cl) and Bromine (Br) based flame retardants meet JS709B low halogen requirements of <=1000ppm threshold. Antimony trioxide based flame retardants must also meet the <=1000ppm threshold requirement.

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

Important Information and Disclaimer: The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

TEXAS INSTRUMENTS

www.ti.com

PACKAGE OPTION ADDENDUM

6-Feb-2020

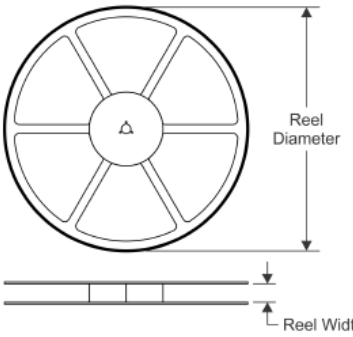
TEXAS INSTRUMENTS

PACKAGE MATERIALS INFORMATION

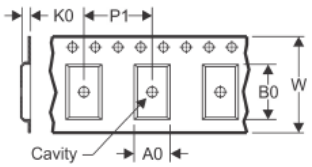
www.ti.com 26-Feb-2019

TAPE AND REEL INFORMATION

REEL DIMENSIONS

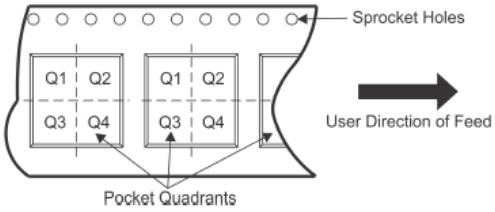


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
DRV8825PWPR	HTSSOP	PWP	28	2000	330.0	16.4	6.9	10.2	1.8	12.0	16.0	Q1

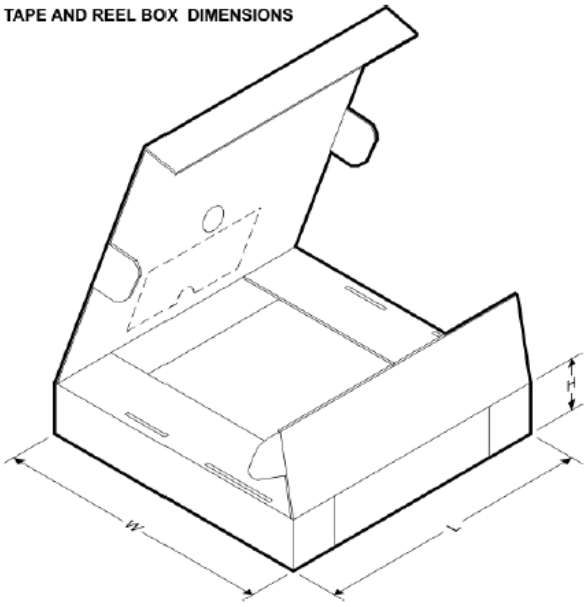
TEXAS INSTRUMENTS

PACKAGE MATERIALS INFORMATION

www.ti.com

26-Feb-2019

TAPE AND REEL BOX DIMENSIONS



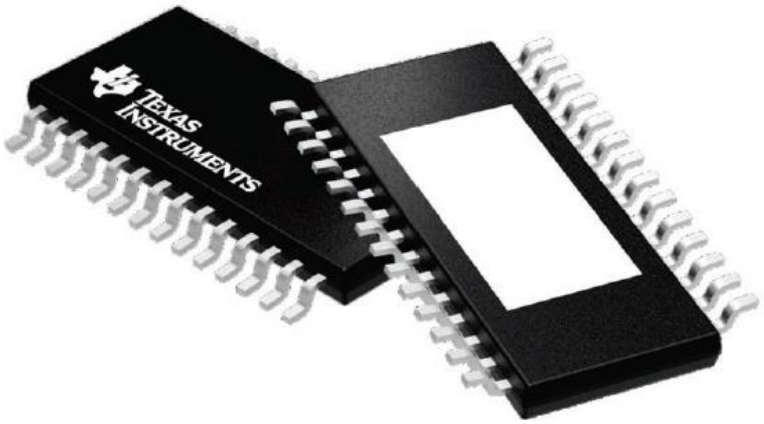
*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
DRV8825PWPR	HTSSOP	PWP	28	2000	350.0	350.0	43.0

GENERIC PACKAGE VIEW

PWP 28 **PowerPAD™ TSSOP - 1.2 mm max height**

4.4 x 9.7, 0.65 mm pitch SMALL OUTLINE PACKAGE



Images above are just a representation of the package family, actual package may vary.
Refer to the product data sheet for package details.

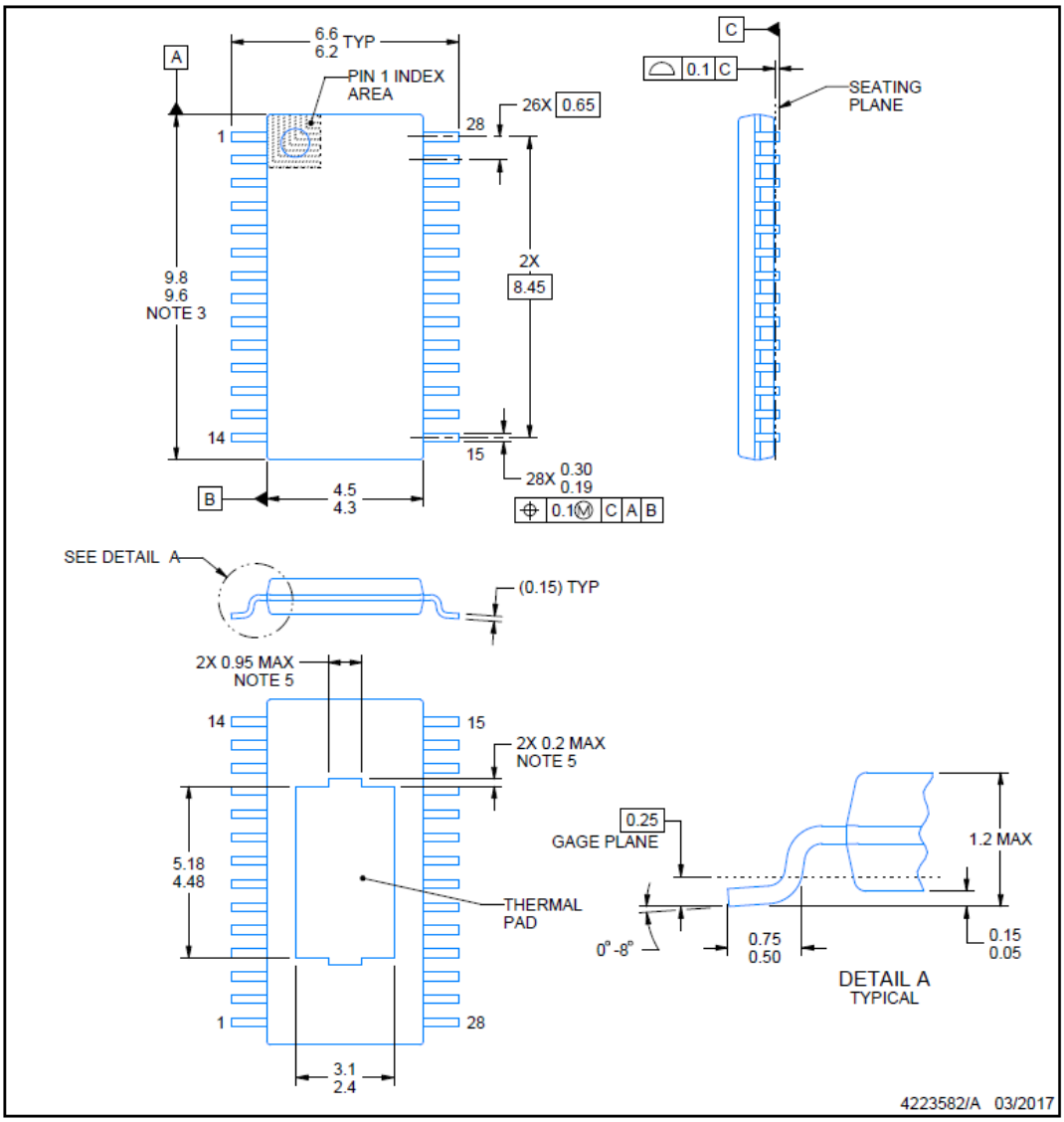
4224765/A

TEXAS INSTRUMENTS
www.ti.com

PACKAGE OUTLINE

PWP0028C **PowerPAD™ TSSOP - 1.2 mm max height**

SMALL OUTLINE PACKAGE



PowerPAD is a trademark of Texas Instruments.

NOTES:

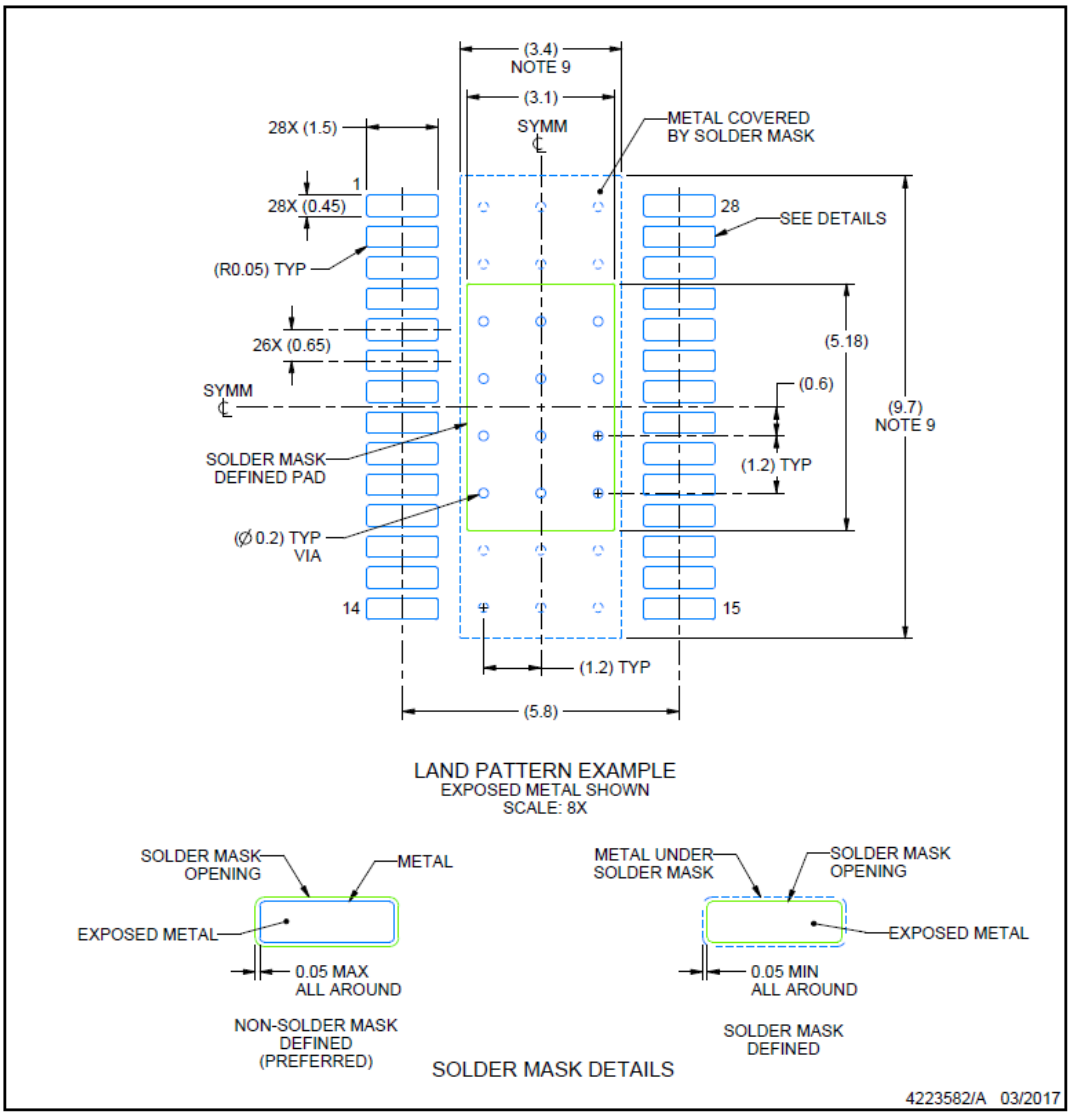
1. All linear dimensions are in millimeters. Any dimensions in parenthesis are for reference only. Dimensioning and tolerancing per ASME Y14.5M.
2. This drawing is subject to change without notice.
3. This dimension does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.15 mm per side.
4. Reference JEDEC registration MO-153.
5. Features may differ or may not be present.

TEXAS INSTRUMENTS
www.ti.com

EXAMPLE BOARD LAYOUT

PWP0028C **PowerPAD™ TSSOP - 1.2 mm max height**

SMALL OUTLINE PACKAGE



NOTES: (continued)

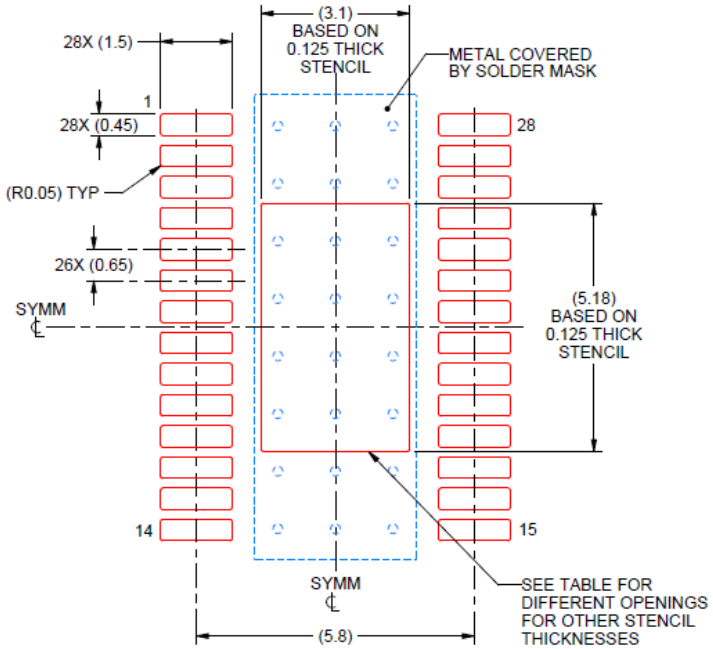
6. Publication IPC-7351 may have alternate designs.
7. Solder mask tolerances between and around signal pads can vary based on board fabrication site.
8. This package is designed to be soldered to a thermal pad on the board. For more information, see Texas Instruments literature numbers SLMA002 (www.ti.com/lit/slma002) and SLMA004 (www.ti.com/lit/slma004).
9. Size of metal pad may vary due to creepage requirement.
10. Vias are optional depending on application, refer to device data sheet. It is recommended that vias under paste be filled, plugged or tented.

TEXAS INSTRUMENTS
www.ti.com

EXAMPLE STENCIL DESIGN

PWP0028C **PowerPAD™ TSSOP - 1.2 mm max height**

SMALL OUTLINE PACKAGE



SOLDER PASTE EXAMPLE
BASED ON 0.125 mm THICK STENCIL
SCALE: 8X

STENCIL THICKNESS	SOLDER STENCIL OPENING
0.1	3.47 X 5.79
0.125	3.10 X 5.18 (SHOWN)
0.15	2.83 X 4.73
0.175	2.62 X 4.38

4223582/A 03/2017

NOTES: (continued)

11. Laser cutting apertures with trapezoidal walls and rounded corners may offer better paste release. IPC-7525 may have alternate design recommendations.
12. Board assembly site may have different recommendations for stencil design.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Anexo 3

Hoja de características del láser de Adafruit

Laser Diode – 5mW 650nm Red

PRODUCT ID: 1054

DESCRIPTION

You have some frickin' sharks at your lair, and you're thinking "it would be really great if I could attach some lasers to them" and then you see these little laser diode modules: what great timing! These encapsulated laser diodes are Class IIIa 5mW, with a 650nm red wavelength. They can be driven from 2.8V to 5.2V so they're great for your embedded electronics project. You can use these for laser harps, electronic 'trip wires', laser-vision guidance, and more! Simply connect power to the red and black wires and you will be lasing in no time.

Please note: this is not a laser pointer. It's a diode with integrated driver. You still need a 3 to 5VDC power supply. These diodes are 5mW output, and have a safety label marked on them. For more information on laser safety and ratings check out the Wikipedia page

Technical Details

RoHS

Weight: 6.3g
Diameter: 10mm / 0.4"
Length: 31mm / 1.2"
Wire length: at least 100mm / 3.9"
650nm wavelength
2.8 – 5.2V DC voltage input
25mA max current draw
Operating temperature –10°C to 40°C;