UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO TEXCOCO

PLATAFORMA AUTOMATIZADA PARA CAPTURA Y MODELADO 3D

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA
ENRIQUE SOLIS CIELO

TUTOR ACADÉMICO
DR. JOEL AYALA DE LA VEGA

TUTORES ADJUNTOS
DR. ALFONSO ZARCO HIDALGO
DR. JAIR CERVANTES CANALES

TEXCOCO, ESTADO DE MÉXICO NOVIEMBRE 2017

Universidad Autónoma del Estado de México
Centro Universitario UAEM Texcoco

DICTÁMEN DE AUTORIZACIÓN Y OBTENCIÓN DE GRADO DE MAESTRÍA

Texcoco, Méx. , a 17 de Noviembre del 2017

Título del proyecto:

PLATAFORMA AUTOMATIZADA PARA CAPTURA Y MODELADO 3D

Tesista:

ENRIQUE SOLIS CIELO

Dictamen: **No. de revisión:** 3

- ☐ **Rechazado**
- ☐ **Sujeto a modificaciones**
- ☐ **Aceptado, condicionado**
- [x] **Aceptado**

Observaciones generales:

Aceptado para la impresión
Aceptado para la defensa de grado

Tutor Adjunto	Tutor Académico	Tutor Adjunto
Dr. Alfonso Zarco Hidalgo	Dr. en Ed. Joel Ayala de la Vega	Dr en C. Jair Cervantes Canales

Centro Universitario UAEM Texcoco
Av. Jardín Zumpango s/n. Fracc. El Tejocote
C.P. 56259 Texcoco, Estado de México.
Tels. (595) 9211216 - 9211247 - 9210368 - 9210493
e-mail: cutex.uaem@gmail.com.
CUTex

Contenido

Índice de Figuras

Ejemplos presentes en la tesis

Resumen

Este trabajo presenta una plataforma automatizada integrada por un módulo de hardware libre y un módulo de software que trabajando en conjunto componen una herramienta tecnológica que permite la captura de imágenes de un objeto y se encarga de manera autónoma de su posterior procesamiento para llevar a cabo un modelado en 3 dimensiones (3D) del mismo.

Se dio prioridad a cuatro puntos esenciales de desarrollo durante la planeación del prototipo:

- El uso de hardware libre debido a la amplia gama de alternativas y fuentes de información referente a estos dispositivos.
- A la revisión del software tanto comercial como libre.
- A la reducción de costos, con el objetivo primordial de asegurar que este prototipo pueda ser desarrollado o adquirido por cualquier institución u organización ya sea de investigación o desarrollo de la forma más accesible
- A la portabilidad, después de un amplio análisis y prueba de diferentes estructuras posibles se optó por una que facilitara la tarea de transportar el aparato.

A lo largo de este documento se pretende abordar el proceso de diseño y desarrollo de esta plataforma iniciando con una breve introducción que nos habla acerca del concepto, además de algunas técnicas y un par de aplicaciones del modelado 3D y algunas otras definiciones de interés para comprender el alcance del proyecto.

Posteriormente se describe la metodología base que se utilizó para agilizar el diseño de este proyecto en general y cada una de las fases de desarrollo de ambos módulos de manera individual, así como las dificultades que se presentaron durante la etapa de diseño de cada uno de los antes mencionados.

Finalmente se describe el funcionamiento completo del proyecto una vez integradas ambas partes del mismo, así como se explican algunas de las pruebas llevadas a cabo sobre este

con la finalidad de comprobar su funcionamiento y rendimiento en un entorno real, además de verificar que la factibilidad de uso respecto al usuario sea la esperada.

Capítulo 1 Introducción

Antecedentes

En la actualidad, con el auge de las tecnologías computacionales y el tremendo avance tecnológico que tenemos a nuestro alcance, es mucho más sencillo diseñar y desarrollar herramientas que hace tiempo solo podían ser desarrolladas por equipos de investigación en organizaciones de alto nivel.

Las computadoras, los dispositivos móviles y hasta las redes sociales han alcanzado un nivel tecnológico que hasta hace un par de décadas solo eran parte de la ciencia ficción.

Actualmente la sociedad ha adoptado una cultura de innovación tecnológica que ha permitido que el uso de las tecnologías de la información y las comunicaciones (TIC's) se conviertan en parte esencial de nuestras vidas tanto en el ámbito social, como cultural y profesional.

En 2016 el Instituto Nacional de Estadística y Geografía (INEGI) en colaboración con la Secretaria de Comunicaciones y Transportes (SCT) y el Instituto Federal de Telecomunicaciones (IFT) revelo mediante la "Encuesta nacional sobre disponibilidad y uso de tecnologías de la información en los hogares, 2015" (INEGI, Instituto Nacional de Estadística y Geografía, 2016) que para el segundo trimestre de dicho año en México al menos 55.7 millones de personas, es decir el 51.3 % de la población (Figura 1.1), tienen acceso a una computadora y 62.4 millones, es decir el 59.5% de la población, son usuarios con acceso a internet (Figura 1.2).

Figura 1.1 Porcentaje de la población en México que son usuarios de computadora.

Fuente: (INEGI, TIC´s en hogares, 2017)

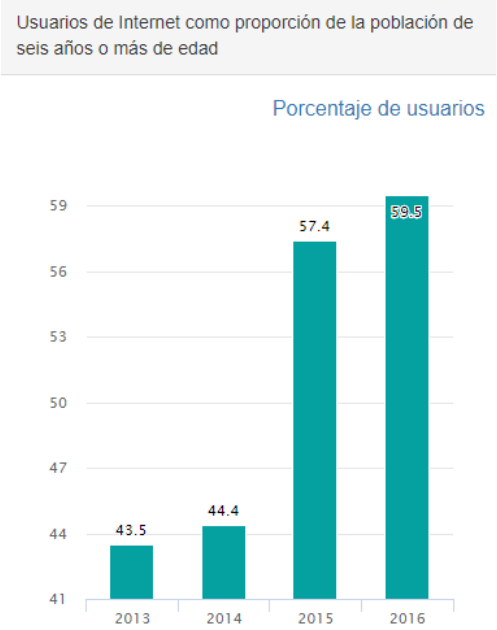

Figura 1.2. Porcentaje de la población en México que son usuarios de Internet.

Fuente: (INEGI, TIC´s en hogares, 2017)

Este rápido avance nos ha permitido desarrollar herramientas tecnológicas de alto alcance desde la comodidad de nuestros hogares, uno de ellos es la técnica de modelado en 3 dimensiones (3D). La importancia del modelado 3D recae principalmente en que es una tecnología tan versátil que puede ser aplicada a un amplio abanico de áreas. Por mencionar algunas podemos considerar el modelado de edificios y estructuras arquitectónicas, el modelado de piezas y prototipos en la industria, el modelado digital de órganos del cuerpo en medicina, la animación en la industria del cine y los videojuegos, la digitalización de piezas históricas, entre muchos otros.

De manera comercial existen muchas organizaciones que proveen herramientas que permiten llevar a cabo las tareas de modelado de manera sencilla. Sin embargo, la mayoría de estas impone ciertas limitantes que para algunos usuarios significan una restricción al uso de las mismas.

Algunas de estas limitantes son el capital económico para adquirir licencias y equipos que lleven a cabo estas tareas, otra es la capacitación que requieren los usuarios para explotar al máximo estas herramientas, por otro lado, si consideramos las alternativas de procesamiento que ofrecen la posibilidad de realizar el trabajo de manera independiente únicamente aportando las imágenes del objeto deseado por medio del internet, tendríamos también que considerar las limitantes en cuanto a ancho de banda, disponibilidad de la red así como disponibilidad de los servidores y tiempo de respuesta de los mismos para llevar a cabo un solo modelado.

Planteamiento del Problema

Desde el punto de vista del software algunas herramientas permiten modelar desde cero y a manera de "dibujo" objetos que podemos encontrar en el mundo real y generarlos de manera digital según nuestra creatividad lo permita, otros realizan el modelado a partir de técnicas de adquisición y digitalización de información del objeto para su posterior procesamiento.

Sin embargo, estas alternativas en su mayoría suelen tener costos elevados de adquisición en cuanto a las licencias de uso y generalmente suelen presentar diferentes inconvenientes técnicos al momento de intentar realizar un modelado.

Tomemos como ejemplo el software de la compañía Autodesk® conocido como ReMake®, este software al igual que su antecesor 123DCatch® (Cline, 2014) promete, a partir de fotografías, generar un modelo 3D de manera sencilla y eficiente para cualquier usuario, esto sin embargo supone ciertas restricciones una vez que se pone a prueba.

La primera restricción con la que nos encontramos, si planeamos utilizar la propuesta de Autodesk, es la licencia de uso. Esta licencia puede ser obtenida desde la página oficial de Autodesk:
https://www.autodesk.com/store/products/remake?term=1year&support=advanced

por un costo mensual de $30 dlls ($534 M/N aproximadamente) o por un costo anual de $300 dlls ($5,334 M/N aproximadamente) tal como se muestra en la Figura 1.3.

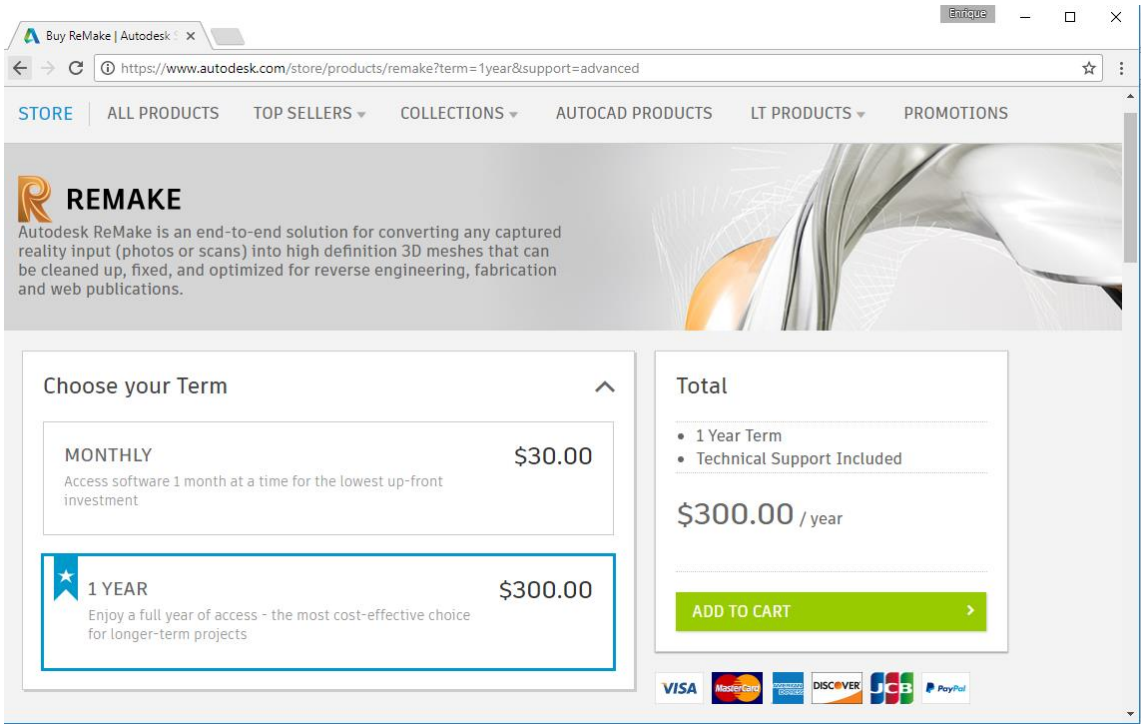

Figura 1.3 Tienda en línea para adquisición de la licencia de ReMake.

Fuente: (Autodesk, 2015).

Cabe mencionar que ReMake cuenta con un periodo de prueba de 15 días (ver Figura 1.4).

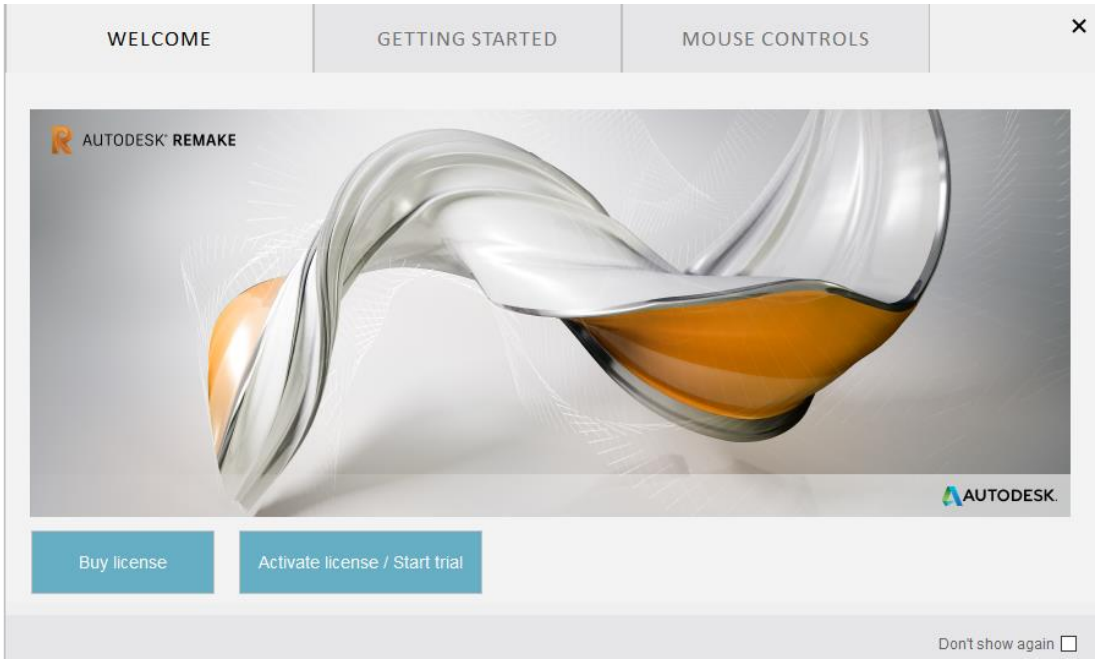

Figura 1.4 Pantalla de activación de licencia o periodo de prueba.

Fuente: (Autodesk, 2015).

La segunda restricción es el conjunto de fotografías, Autodesk® recomienda un conjunto de entre 20 a 40 fotografías en resolución estándar de 5 MP, las cuales deben ser tomadas desde diferentes ángulos alrededor del objeto a distancias relativamente equitativas, inicialmente esto no suena complicado. Sin embargo, tomando en cuenta que la tarea de captura se realiza de manera manual, entonces esta se complica. Por otro lado ReMake® ofrece la posibilidad de procesar estas imágenes de dos maneras (Figura 1.5), de manera local siempre y cuando la computadora en la que se esté trabajando cuente con una tarjeta para gráficos de la marca Nvidia® y al menos 16 GB en RAM (Figura 1.6). A la par también se puede llevar a cabo la tarea de modelado en línea mediante alguno de los servidores de Autodesk, el proceso requiere subir vía Internet el conjunto de fotografías mediante la misma aplicación y esperar a que la misma aplicación indique que el proceso de envió y digitalización está completo. Aunado a esto debemos considerar que el procesamiento online no está exento de requerimientos mínimos, entre los cuales podemos destacar la necesidad de contar con un mínimo de 4GB de RAM, un procesador multi núcleo y una tarjeta de gráficos integrados como mínimo, además de la obvia necesidad de una buena conexión a internet (Figura 1.7).

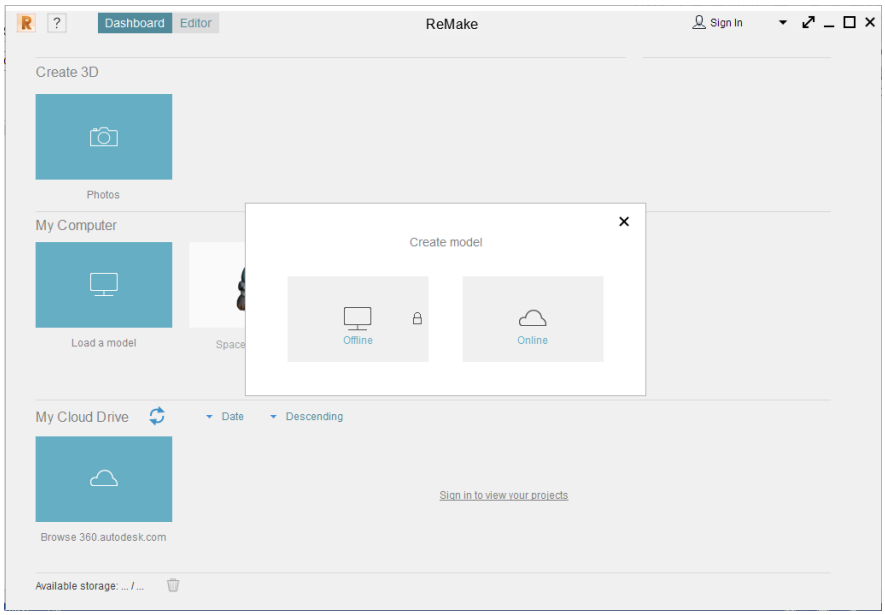

Figura 1.5. Software de Modelado ReMake.

. *Fuente*: Elaboración propia.

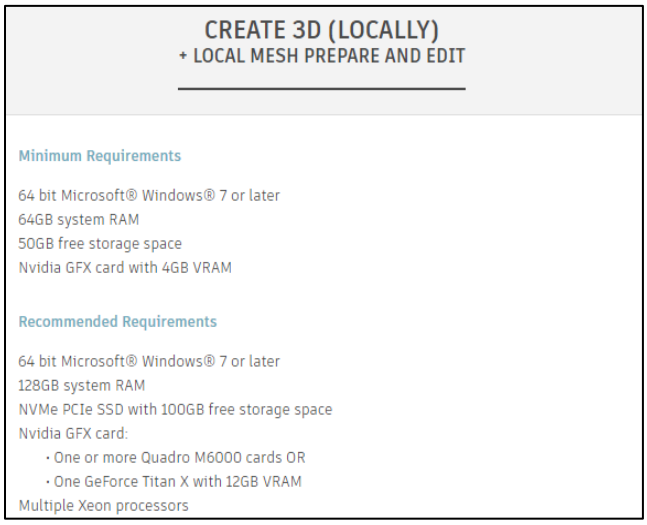

Figura 1.6. Lista de requerimientos para procesamiento local en ReMake.

Fuente: (Autodesk, 2015)

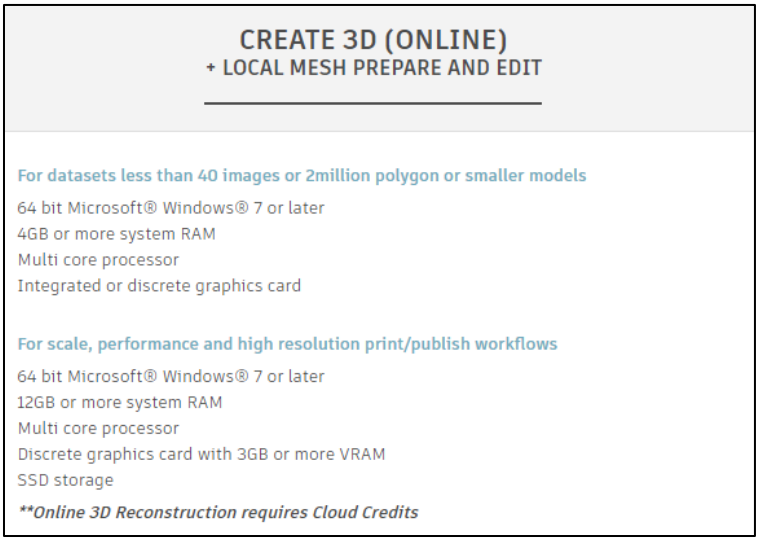

Figura 1.7. Lista de requerimientos para procesamiento remoto en ReMake.

Fuente: (Autodesk, 2015)

Otra alternativa, como software de modelado es PhotoScan®, producto de la compañía Agisoft®, PhotoScan®, es un avanzado software de modelado 3D que funciona de manera local en el computador y genera una representación digital de un objeto a partir de solo fotografías del mismo. Al igual que ReMake®, PhotoScan® es un software de paga con un periodo de prueba de 30 días, posterior a este periodo la licencia del software puede ser adquirida desde $3,205 M.N, o se puede adquirir una licencia especial comprobando los fines educativos de la misma desde $1,056 M.N mediante la página oficial http://www.agisoft.com/buy/online-store/ (Figura 1.8).

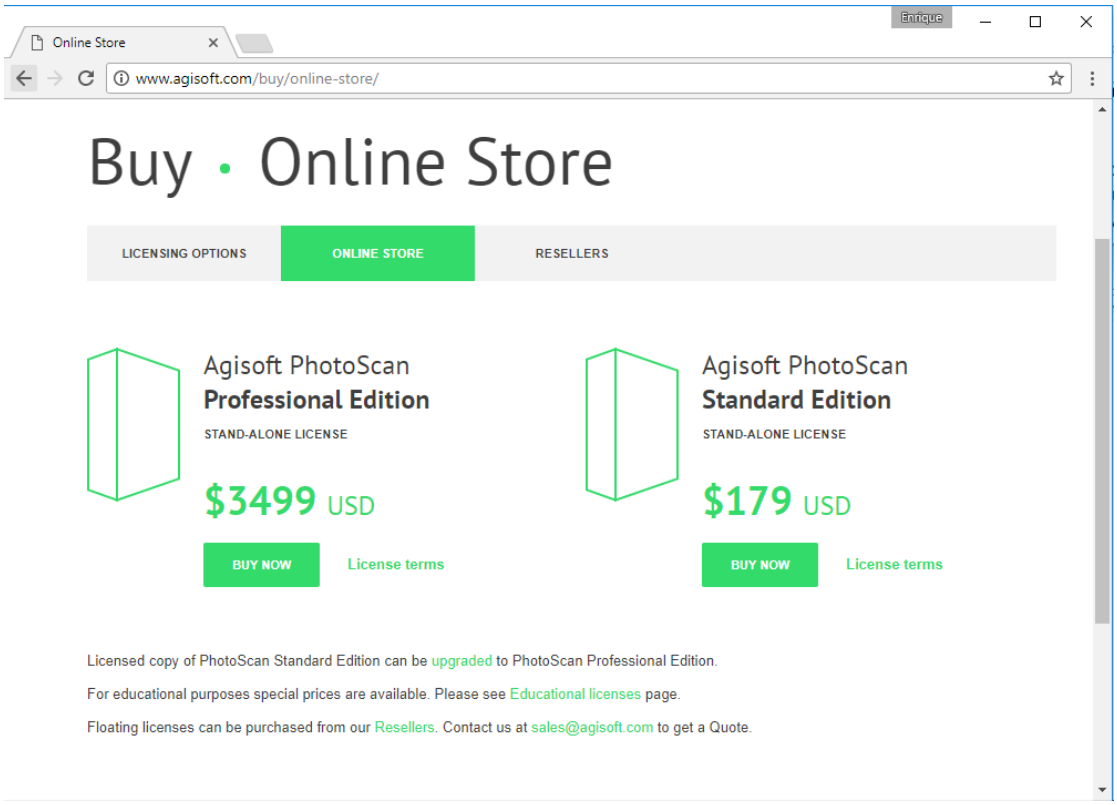

Figura 1.8. Tienda online del Software de modelado PhotoScan.

Fuente: (Agisoft, 2017)

Al igual que con el software, también se pueden encontrar de manera comercial diferentes alternativas de Hardware destinados a llevar a cabo la tarea del modelado. Algunos de los ejemplos más económicos que podemos encontrar a la venta son el escáner de mano Sense (Figura 1.9) desarrollado por la compañía 3D Systems con un precio de $15,000 M.N y el escáner 3D “Matter and Form 3D” (Figura 1.10) de la empresa Matter and Form Inc. con un precio estimado de $22,000 M.N.

Figura 1.9. Escaner de mano Sense.

Figura 1.10. Escaner Matter and Form 3D.

Tomando todo lo anterior en cuenta podemos deducir que para algunas organizaciones, como las educativas, no es completamente rentable adquirir un dispositivo de este tipo debido a su alto costo y que además requeriría de cierto nivel de conocimientos computacionales para poder explotar sus características al máximo, o por otro lado, de un costo extra por capacitación para el uso del dispositivo.

Objetivo General

Diseñar, desarrollar e implementar una plataforma compuesta por un módulo de software y un módulo de Hardware libre que permita automatizar la tarea de modelado en tres dimensiones de objetos del mundo real a partir de fotografías del mismo.

Objetivos Específicos

1. Diseñar y desarrollar un módulo de hardware libre que permita la automatización del proceso de captura de imágenes de un objeto a modelar.
2. Buscar e implementar una herramienta de software que permita, a partir de un conjunto de imágenes, generar un modelo en tres dimensiones de un objeto real.
3. Diseñar una interfaz gráfica de usuario atractiva e intuitiva que asegure la factibilidad de uso para cualquier persona.
4. Integrar ambos módulos en una sola plataforma autónoma que oculte al usuario final la complejidad de procesamiento.
5. Validar la eficacia y funcionalidad de la plataforma integrada.

Supuesto

Este proyecto proveerá una herramienta tecnológica que permita generar modelos en 3 dimensiones (3D) de objetos y aplicable a un amplio abanico de áreas de la ciencia a un costo considerablemente más accesible en comparación con las alternativas comerciales habituales, y a su vez ofrecerá un modelado de calidad para posteriores análisis y tratamientos de los objetos generados a partir del escaneo y su consecuente procesamiento digital. Además de esto, la interfaz de usuario final que sea generada será tan intuitiva que permitirá que casi cualquier persona pueda hacer uso de ésta, aun cuando no se cuente con previa capacitación en cuanto al área de cómputo se trata.

Justificación

Un reto interesante es el uso del hardware y el software para hacer 3D que sea económico y de fácil uso. Considerando la amplia gama de hardware libre, como puede ser Arduino o Raspberry (Figura 1.11), con la que podemos complementar la integración de plataformas tecnológicas que nos permitan procesar en tiempo real la estructura de objetos con la finalidad de generar un modelo 3D de los mismos, podemos deducir que la posibilidad de generar proyectos con dicha capacidad de manera más accesible se eleva significativamente eliminando algunas de las principales restricciones que suponen la adquisición de alternativas de Software o Hardware que se encuentran actualmente en el mercado.

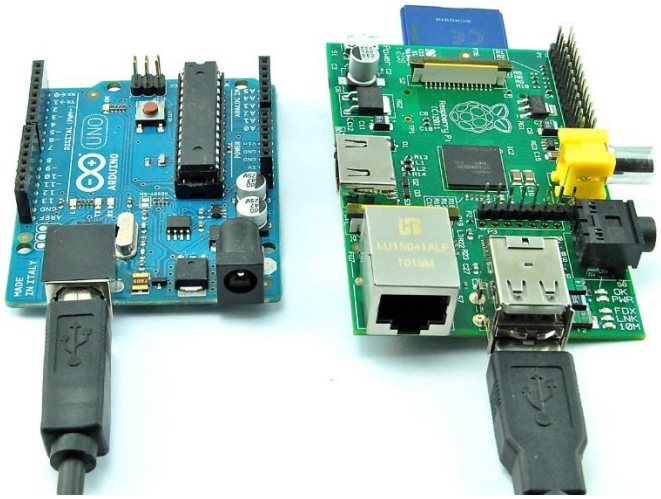

Figura 1.11. Plataformas de Hardware Libre Arduino UNO y Raspberry Pi.

Dado que el modelado 3D tiene un sinfín de aplicaciones, el desarrollo de alternativas tecnológicas como la presentada en este documento permitirá a muchos campos adaptarse y explotar las nuevas tecnologías que puedan impulsarlos en la carrera tecnológica en la que vivimos actualmente y de esta manera contribuir a que cada una de ellas se posicione de manera mucho más competitiva en esta.

Tomemos como ejemplo el caso de disciplinas como la Entomología cuyo material principal de estudio son los insectos, un entomólogo necesita estudiar y analizar continuamente un ejemplar de alguna de las miles familias de estas formas de vida por periodos prolongados de tiempo. Sin embargo, estos ejemplares son demasiado frágiles como para manipularlos y están expuestos a sufrir deterioro o destrucción total si no se manejan con la debida precaución. Teniendo un modelo digital de cada uno de estos ejemplares su tratamiento y análisis se simplifica en gran medida y supone para el investigador la posibilidad de generar contenido digital mucho más completo de cada espécimen.

Así pues, esta propuesta puede ser enfocada a un gran número de disciplinas y proveer así de una herramienta innovadora que les simplifique en gran medida sus labores.

Capítulo 2 Marco Teórico

Visión Artificial

Como disciplina científica la visión por computador es aquella que se ocupa de la teoría y la tecnología para construir sistemas artificiales que obtienen información a partir de imágenes, o datos multidimensionales. Dado que la percepción puede ser vista como la extracción de información de señales sensoriales la visión por computador puede ser vista como la investigación científica de sistemas artificiales para la percepción a partir de imágenes o datos multidimensionales (ISLab, 2017). Además, esta disciplina está estrechamente relacionada con otras tales como el procesamiento de imágenes y la visión máquina. Disciplinas también enfocadas a la obtención y manipulación de datos a partir de imágenes (Figura 2.1).

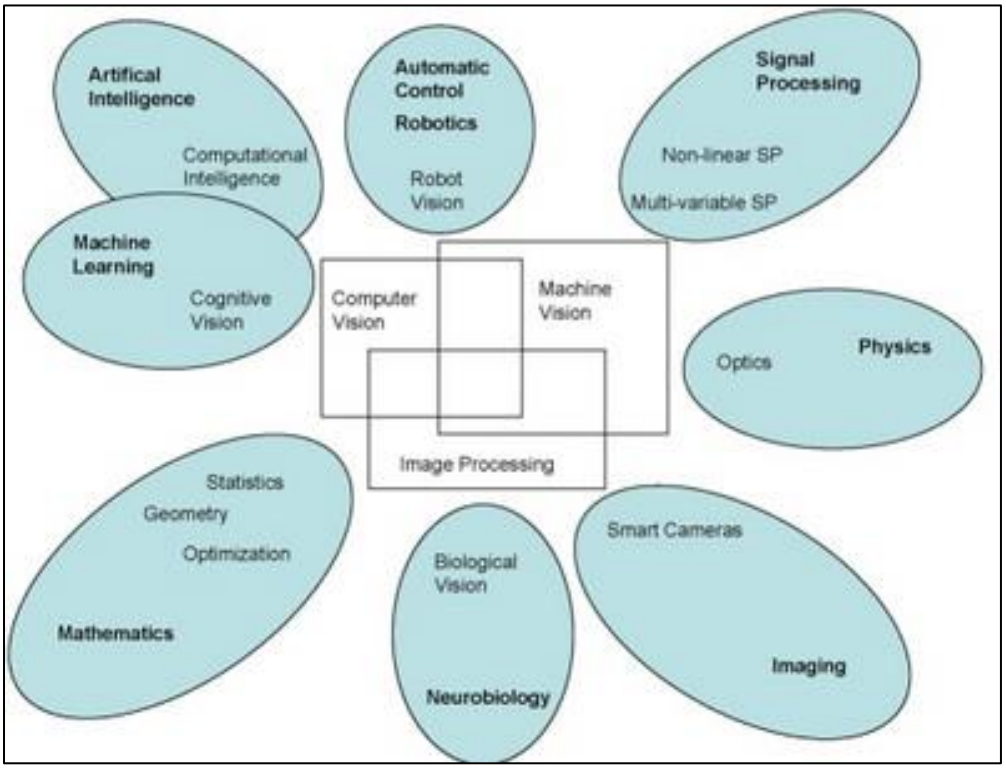

Figura 2.1. Disciplinas relacionadas a la visión por computador.

Fuente: (ISLab, 2017)

Durante las últimas tres décadas muchos de los trabajos realizados en cuanto al área de reconocimientos de patrones y una significativa fracción de los trabajos en inteligencia

artificial han sido enfocados al análisis y la interpretación de imágenes. Las aplicaciones de estos métodos tienen una amplia gama de aplicaciones como el reconocimiento de patrones, reconocimiento de caracteres, microscopia, radiología, automatización industrial, robótica, censado remoto, navegación, reconstrucción y reconocimiento, por mencionar algunas de las más importantes (Martin A. Fischler, 2014).

Un sistema de visión por computador o visión artificial está compuesto esencialmente por dos partes fundamentales: El sistema de adquisición de imágenes (Hardware) y el sistema de tratamiento de las imágenes (Software) (Somolinos Sánchez, 2002).

Dentro del primero, la estructura básica de un sistema de visión por computador (representada en la Figura 2.2, donde T.A.I. se refiere a la tarjeta de adquisición de imagen, el C.P.U. es la unidad central de procesamiento, el T.C. es la tarjeta de control y la T.I.A.D es la tarjeta de interface analógico-digital) está compuesta por el propio objeto o la escena, el sensor o cámara empleada, el sistema de iluminación y el computador central.

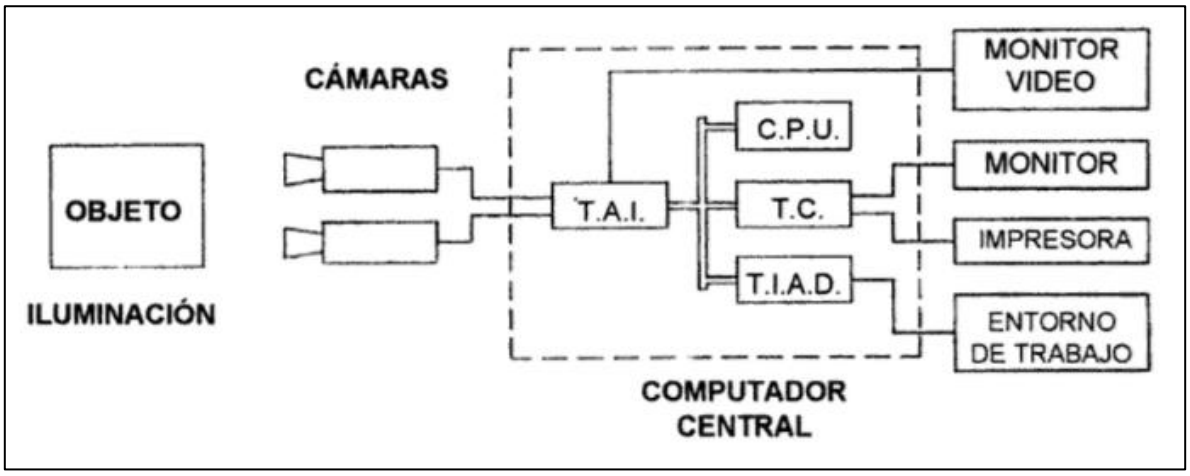

Figura 2.2. Estructura básica de un sistema de visión por computador.

Fuente: (Somolinos Sánchez, 2002)

En cuanto al sistema de tratamiento de imágenes, éste constituye en definitiva el software del sistema que se ejecuta en el hardware para llevar a cabo su objetivo el cual es esencialmente la interpretación automática de una escena tridimensional a partir de las imágenes tomadas de la misma. En éste sistema se puede consideran las etapas básicas de tratamiento de imágenes tales como el procesamiento, la segmentación y el reconocimiento (Somolinos Sánchez, 2002).

Modelado 3D

El modelado en tres dimensiones (3D) es la técnica mediante la cual se busca reproducir en la memoria de un computador objetos del mundo real, manteniendo fielmente sus características físicas tales como sus dimensiones, volumen y forma. Además existen dentro de la visión artificial una gran variedad de métodos cuyo objetivo principal es obtener un algoritmo capaz de realizar la conexión del conjunto de puntos representativos del objeto en forma de elementos de superficie (Mesa Múnera, Ramírez Salazar, & Branch Bedoya, 2010).

El interés en el modelado 3D surge debido a que esta tecnología tiene aplicaciones en una amplia variedad de campos, a saber: diseño, automatización de manufactura, mapeo de terrenos, conducción autónoma, arqueología, vigilancia, robots inteligentes, entre muchos otros.

Los métodos de reconstrucción 3D son utilizados para convertir una nube de puntos en una representación de superficie. Desafortunadamente, muchos métodos no describen analíticamente el modelo ya que utilizan representaciones que aproximan la superficie del objeto como mallas triangulares.

En general, el proceso de reconstrucción tridimensional involucra las siguientes etapas:

- Adquisición: Esta etapa consiste en el censado de la superficie del objeto a partir de un número determinado de vistas o imágenes.
- Registro: Consiste en emparentar dos o más imágenes que pueden ser tomadas en diferentes momentos desde diferentes sensores o diferentes puntos de vista. El

registro de múltiples imágenes solapadas consiste en llevar todas las imágenes adquiridas a un sistema de coordenadas común, esto puede crear datos redundantes, los cuales pueden ser integrados en un conjunto de puntos menos densos.

- Integración: Una vez registradas las diferentes imágenes, se tiene un conjunto de datos en el espacio tridimensional cuya conectividad y relación aún no están bien definidas. La integración tiene como objetivo disminuir las redundancias y generar datos en regiones con ausencia de información para obtener una representación computacional suave y continua del objeto.
- Ajuste de superficies: El objetivo de esta etapa es brindar la mejor estimación digital de una superficie real descrita por un conjunto conocido de puntos. Este ajuste se concentra en obtener una descripción digital precisa, concisa y aproximada a la superficie real a partir de un modelo matemático. Estos modelos pueden tener una variedad de representaciones, entre las más utilizadas se encuentran las mallas triangulares.

Hardware libre

El hardware libre (también llamado “Open-Source”) o de “fuente abierta) comprende aquellas plataformas y dispositivos que permiten a los usuarios puedan estudiarlos para comprender de manera técnica su funcionamiento, modificarlo, reutilizarlo, mejorarlo y posteriormente compartir dichos cambios con el resto de usuarios. Para conseguir esto, la comunidad debe poder tener acceso a los diseños esquemáticos del diseño del hardware en cuestión. Estos detallan toda la información necesaria para que cualquier usuario con los materiales, herramientas y conocimientos necesarios puedan reconstruir dicho hardware por su cuenta sin problemas, ya que consultando dichos archivos se puede conocer que componentes individuales integran el hardware y el modo de interconexión entre cada uno de ellos (Torrente Artero, 2013).

A diferencia del mundo del Software libre donde el ecosistema de licencias libres es rico y variado, en el ámbito del hardware todavía no existen prácticamente licencias específicamente de hardware libre ya que dicho concepto es relativamente nuevo. Fue hasta 2010 que surgió el proyecto OSHW (Open Source Hardware) (Freedom-Defined, 2017), el cual pretende establecer una colección de principios que ayuden a identificar como "hardware libre" un producto físico. Cabe destacar que OSHW no es una licencia, sino una declaración de intenciones aplicable a cualquier artefacto físico para que pueda ser considerado libre. El objetivo de la OSHW es ofrecer un marco de referencia donde se respete la libertad de los creadores para controlar su tecnología y al mismo tiempo establecer mecanismos adecuados para compartir el conocimiento y fomentar el comercio a través del intercambio abierto de diseños.

Arduino

Arduino es una plataforma de hardware libre basada en un simple tablero de entradas y salidas (I/O) y un ambiente de desarrollo que implementa el lenguaje Processing, el cual es un flexible software y lenguaje de aprendizaje para codificación dentro del contexto de las artes visuales. Desde 2001 Processing ha promovido la incorporación del software dentro de las artes visuales como parte de la tecnología por lo cual lo hace ideal para estudiantes, artistas, diseñadores, investigadores y aficionados en el aprendizaje y desarrollo de software (Fry & Reas, 2017).

Arduino puede ser utilizado para desarrollar objetos interactivos autónomos o puede conectarse vía software con una computadora. La plataforma puede ser ensamblada a mano o puede adquirirse ya funcionando; el IDE (Integrated Development Environment/Entorno de Desarrollo Integrado) puede ser descargado gratis de la página oficial de Arduino: www.arduino.cc (Banzi, 2011).

Algunas características que hacen diferente a Arduino de otras plataformas en el mercado son:

- Cuenta con un entorno de desarrollo multiplataforma: Puede ser ejecutado en Windows, Macintosh y Linux.
- Está basado en el IDE de programación Processing, un entorno de desarrollo usado por artistas y diseñadores.
- Se programa vía USB y no vía puerto serial.
- Se considera como hardware y software libre, el diagrama del dispositivo puede ser adquirido en la página oficial de Arduino.
- Tiene un costo reducido en comparación con otras alternativas (dependiendo de la versión que se desee adquirir).
- Cuenta con una activa comunidad de usuarios.

El proyecto Arduino se inició en el año 2005 en la ciudad de Ivrea, provincia de Turin, Italia, en el Instituto de Interactividad y Diseño, a partir de una idea de los profesores David Cuartielles y Massimo Banzi. El objetivo de éste fue crear una herramienta de hardware que fuese fácilmente programable por usuarios no especializados en ordenadores y a bajo costo. A estos dos profesores se les unieron otros especialistas que desarrollaron un IDE que traduce un lenguaje de alto nivel a lenguaje máquina para que Arduino lo comprenda (Caicedo Pedrera, 2017).

Arduino es un kit de desarrollo capaz de interpretar variables en el entorno y transformarlas en señales eléctricas correspondientes a través de sensores conectados a sus terminales de entrada y tutelar el control o accionamiento de algún otro elemento electrónico conectado a su terminal de salida. Ademas, está basado en los microcontroladores AVR de Atmel, en concreto en los modelos ATmega8, ATmega168, ATmega328 y ATmega1280; según el microcontrolador utilizado Arduino recibe su nombre (Figura 2.3Figura 2.3).

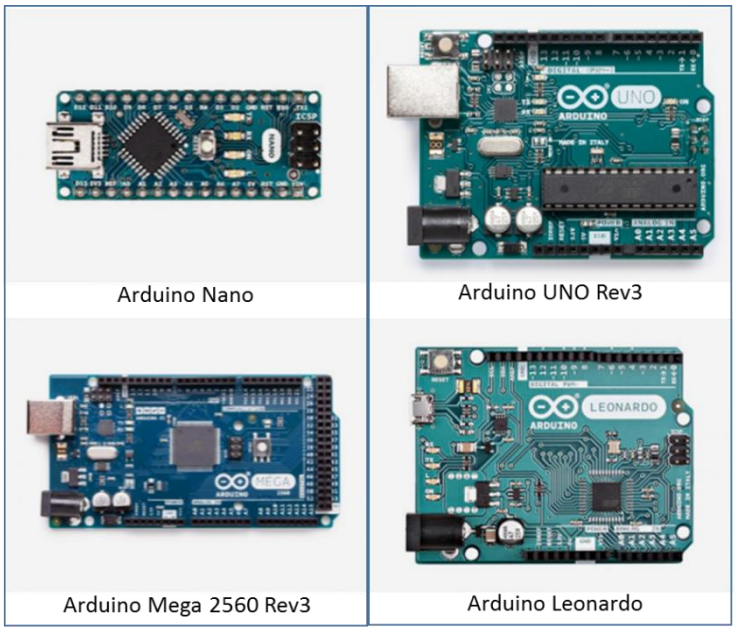

Figura 2.3. Algunas versiones de Arduino con diferentes características.

Fuente: (Arduino, 2017).

Java

Java es un lenguaje de programación de alto nivel orientado a objetos. Fue desarrollado por la empresa Sun MicroSystems a principios de los 90's, y presentado oficialmente en Mayo de 1995 en la conferencia Sun World. Recientemente en 2010, Sun Microsystems fue adquirida por Oracle Corporation.

Java se pensó como una tecnología que permitiese el desarrollo de programas capaces de ejecutarse en entornos distribuidos y heterogéneos, es decir, programas ejecutables en un amplio abanico de plataformas de hardware y software, principalmente dispositivos electrónicos embebidos. La idea principal fue el desarrollo de un sistema operativo portable de pocos recursos, en tiempo real y distribuido, pero finalmente se obtuvo un lenguaje de programación (Garrido Abenza, 2015).

Entre algunas de las características que podemos destacar de Java podemos mencionar:

- Familiar: El lenguaje Java se desarrolló desde cero pero como un nuevo lenguaje pero su sintaxis es muy similar al lenguaje C o C++, por lo que se facilita la migración de aquellos desarrolladores ya familiarizados con dichos lenguajes.
- Sencillo: Aunque Java pueda tener una curva de aprendizaje dura, el conocer otros lenguajes de programación similares facilita su aprendizaje. Además, lo aprendido sirve para el desarrollo en distintos dispositivos para los que tradicionalmente se utilizaban herramientas y lenguajes muy diferentes entre sí.
- Multiplataforma: Java fue diseñado específicamente para ser "Write Once, Run Anywhere" (WORA), es decir, escribir (y compilar) una sola vez en una plataforma, y ejecutar en cualquier otra, sin necesidad de modificar el código fuente ni recompilar. La portabilidad de los programas Java se consigue dado que este es un lenguaje interpretado, aunque también compilado para mejorar el rendimiento de la interpretación. Cuando un programa en Java se compila se genera también un código intermedio independiente a la plataforma denominado bytecode. Estos bytecodes serán los que se distribuyan para su ejecución, siendo interpretados por la Máquina Virtual de Java (JVM) especifica de la plataforma del usuario.
- Alto Rendimiento: Los programas Java no son tan rápidos como los desarrollados con lenguajes que se compilan de forma nativa para una plataforma concreta ya que son interpretados durante su ejecución. Sin embargo no son tan lentos como los exclusivamente interpretados, es decir, los que van interpretando el código fuente línea a línea comprobando en cada caso la sintaxis. Sin embargo si se requiere de mayor velocidad, Java puede integrarse con fragmentos de código nativo de C o ensamblador.
- Robusto: Durante la compilación de un programa java se comprueba la sintaxis y ciertas situaciones que con otros lenguajes compilados no se comprobarían, pudiendo dar lugar a resultados inesperados. Además, el recolector de basura o garbage collector de Java se encarga de administrar la gestión de memoria dinámica.

- Orientado a objetos: Esta metodología de programación suele ser muy flexible y facilita todo el ciclo de vida del software desde el análisis y diseño hasta el mantenimiento.
- Distribuido: Java dispone de una librería de clases que permiten la comunicación entre programas ejecutados en ordenadores remotos conectados en red (distribuidos), en un entorno heterogéneo y de forma segura, esto es, evitando la intrusión de otros programas.
- Concurrente: Java permite el desarrollo de programas concurrentes o multi-hilo para conseguir un mejor rendimiento y aprovechar el procesador cuando sea necesario realizar varias tareas al mismo tiempo.

TIC´s

Las tecnologías de la información y las telecomunicaciones (TIC's) están presentes en todos los niveles de nuestra sociedad actual, desde las grandes corporaciones multinacionales hasta la pequeñas y medianas empresas (PYMES), gobiernos, administraciones, universidades, centros educativos, etc.

Ordenadores, teléfonos celulares, reproductores MP3, tarjetas de memoria, Discos Versátiles Digitales (DVD), Sistemas de Posicionamiento Global (GPS), Internet, etc., son tecnologías que se han convertido imprescindibles para nuestra sociedad.

Para poder definir un concepto de las TIC's iniciemos por definir los elementos que las componen.

Por una parte tenemos el concepto de la tecnología, definida como la ciencia que estudia los medios técnicos y los procesos empleados en las diferentes ramas de la industria y los negocios.

Por otra parte, la tecnología de la información (o informática) es la ciencia que estudia las técnicas y procesos automatizados que actúan sobre los datos y la información. La palabra informática proviene de la fusión de los términos “información” y “automática”.

Además, las tecnologías de la comunicación estudian las técnicas y procesos que permiten el envío y recepción de información entre dispositivos.

Así pues, podemos decir que las TIC’s son aquellas tecnologías en las que un mensaje (instrucciones y datos) se transmiten entre emisor y receptor (usuarios) a través de un canal digital (hardware), establecidos por un código (software) dentro de un contexto establecido por convenios internacionales (Suares y Alonso, 2010).

Actuadores

Los actuadores o accionadores son dispositivos que dotan de movimiento a las articulaciones o piezas móviles de un sistema robótico o electrónico. Suelen contar con un elemento motriz y un elemento de transmisión (Somolinos Sánchez, 2002).

El elemento motriz es un servomecanismo de posición que según su soporte tecnológico puede ser:

- Neumático (Movimiento de traslación)
- Hidráulico (Movimiento de traslación o rotación)
- Eléctrico (Movimiento de rotación). Los accionamientos eléctricos pueden estar constituidos a su vez por:
    - Motores paso a paso
    - Motores de corriente continua
    - Motores de corriente alterna

Sensores

Un sensor es un dispositivo de entrada que sirve como intermediario entre una variable física y un sistema. Estos entregan señales eléctricas a la salida, ya sean analógicas o

digitales, debido a que este tipo de dominio es el más utilizado en los sistemas de medición actuales.

Los sensores pueden clasificarse de muchas formas distintas, pero las más comunes son por el tipo de variable a medir o por el principio de transducción utilizado en ellos.

Tomando en cuenta la clasificación según el tipo de variable medida, que suele ser la más común, podemos dividirnos como se muestra a continuación en la Figura 2.4:

Fuente: (Corona Ramírez, Abarca Jiménez, & Mares Carreño, 2014).

Clasificación de los sensores según la variable física a medir	De posición, velocidad y aceleración
	De nivel y proximidad
	De humedad y temperatura
	De fuerza y deformación
	De flujo y presión
	De color, luz y visión
	De gas y pH
	Biométricos
	De corriente

Figura 2.4. Clasificación de los sensores según el tipo de variable medida.

Todos los sensores, sin importar cuál sea su principio de transducción o el tipo de variable física que midan, siempre tienen características particulares que los distinguen. Entre algunas de ellas podemos encontrar:

- Sensitividad: La entrada mínima que requiere este para provocar una salida estable.
- Rango: El intervalo presente entre el valor mínimo y el valor máximo de la variable que puede medir el sensor.
- Precisión: El grado de repetitividad de una medida.
- Exactitud: La diferencia máxima entre la salida actual del sensor y el valor real de la variable medida.
- Tiempo de respuesta: El periodo que transcurre desde que la variable censada presenta un cambio de estado y el sensor lo registra.

Capítulo 3 Materiales y Métodos

Materiales

Para llevar a cabo esta propuesta se utilizó una computadora portátil de la marca Dell con un sistema operativo Windows 10 Pro, un procesador Intel Core i7-5500U a 2.4 Hz y 8 GB de RAM. Además de una tarjeta gráfica NVIDIA GeForce 920M.

En cuanto a la captura de imágenes se utilizó la cámara web usb Ele-gate HD de la marca Ele-Gate a la cual se le desmonto la cubierta y se le retiro el pequeño sistema de iluminación led con el que venía originalmente para reducir peso y tamaño y permitir una mejor manipulación de la misma.

Para llevar a cabo la tarea de automatización se utilizó la plataforma Arduino en su modelo Mega 2560 Rev3 (Figura 3.1). Este modelo permite el intercambio de señales eléctricas mediante sus 16 entradas analógicas y sus 54 pines digitales de entrada/salida. Además, cuenta con un microcontrolador ATmega2560 y una memoria flash de 256k; este se conecta de manera serial vía USB a la computadora.

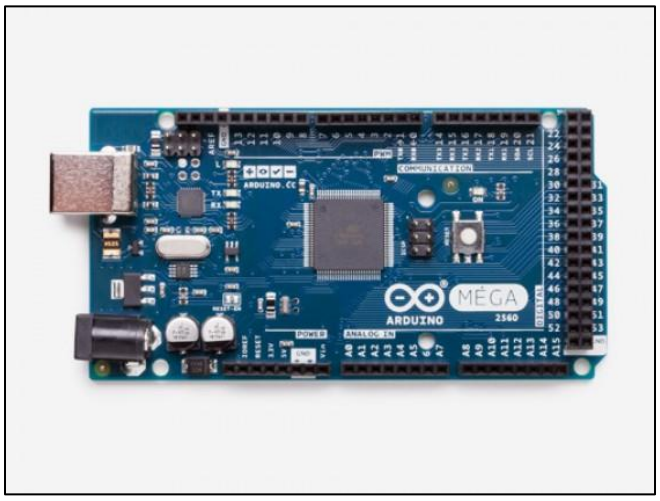

Figura 3.1. Arduino Mega 2560 Rev3.

Fuente: (Arduino, 2017).

Arduino provee de su propio editor para llevar a cabo la programación del microcontrolador el cual es distribuido de manera gratuita en la página oficial de Arduino. Para el desarrollo de este proyecto se utilizó la versión 1.6.5.

En la Figura 3.2, se muestra la interfaz principal del editor de Arduino, el cual cuenta con las siguientes características (Enriquez Herrador, 2009):

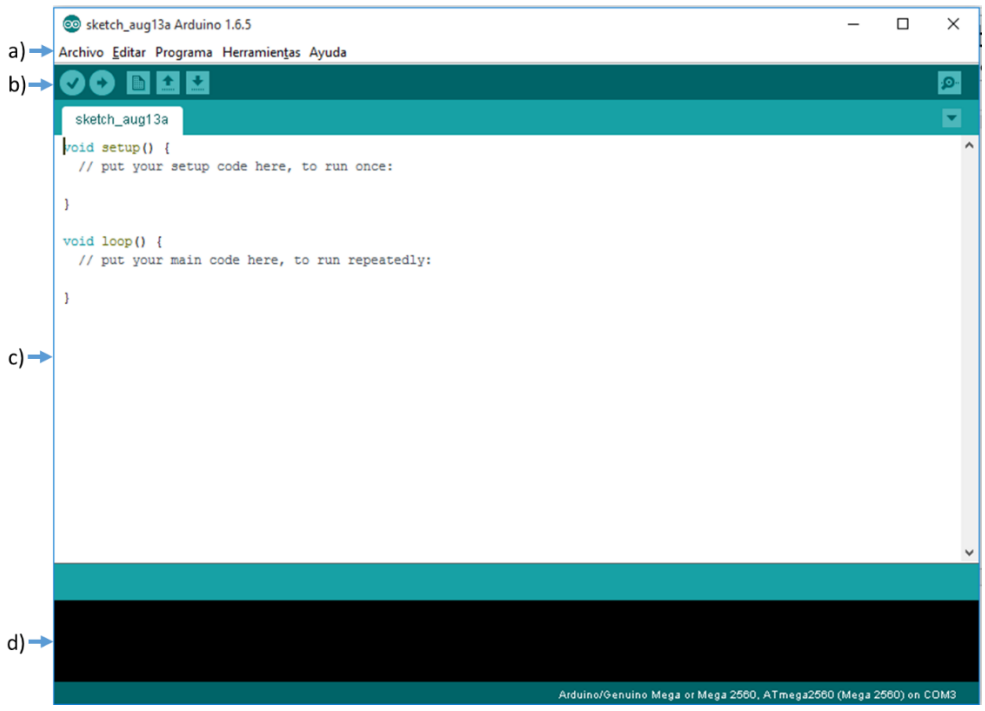

Figura 3.2. Interfaz principal del editor de Arduino.

Fuente: Elaboración propia.

- a) Barra de menús.
- b) Verificar (revisa la sintaxis del script), Subir (se encarga de grabar el script en la memoria del microcontrolador), Nuevo (crea un nuevo script), Abrir (Busca y carga al editor un script previamente creado), Salvar (guarda el script actual) y Monitor serie (Abre la ventana de monitoreo de envío y recepción de datos en serie).
- c) Área de edición.
- d) Monitor de compilación.

En cuanto al sistema de movimiento se emplearon dos motores a pasos 28BYJ-48 (Figura 3.3), cada uno con su correspondiente driver del tipo ULN2003A (Figura 3.4). Estos motores son habituales como piezas clave en la enseñanza de pequeños robots y posicionadores caseros programados con Arduino. Las características técnicas con las que dicho motor cuenta son (Prometec, 2017):

- Tensión nominal de entre 5V y 12V.
- 4 Fases de excitación.
- Resistencia de 50Ω.
- Par motor de 34 N/m.
- Consumo aproximado de 55 mA.
- 8 Pasos por vuelta.
- Reductor de 1/64.

Traduciendo los dos últimos de manera más comprensible podemos deducir que multiplicando el número de pasos por vuelta (8) por el índice de reducción (64) nos da como resultado un total de 512 impulsos eléctricos para completar una vuelta completa del motor.

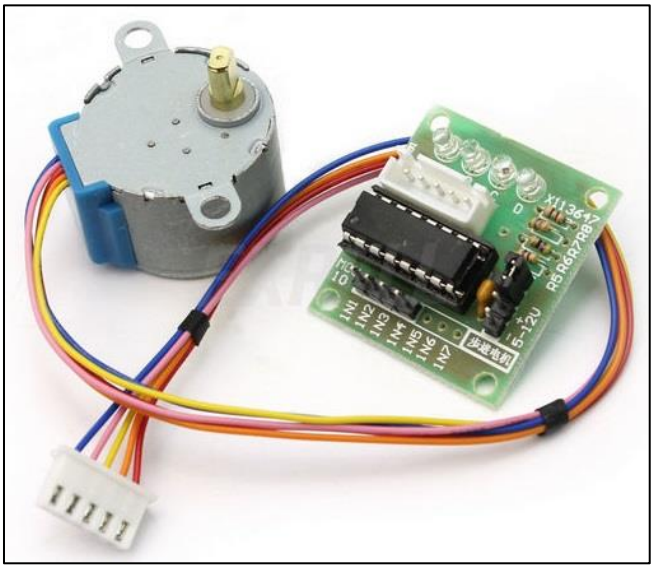

Figura 3.3. Motor a pasos unipolar 28BYJ-48.

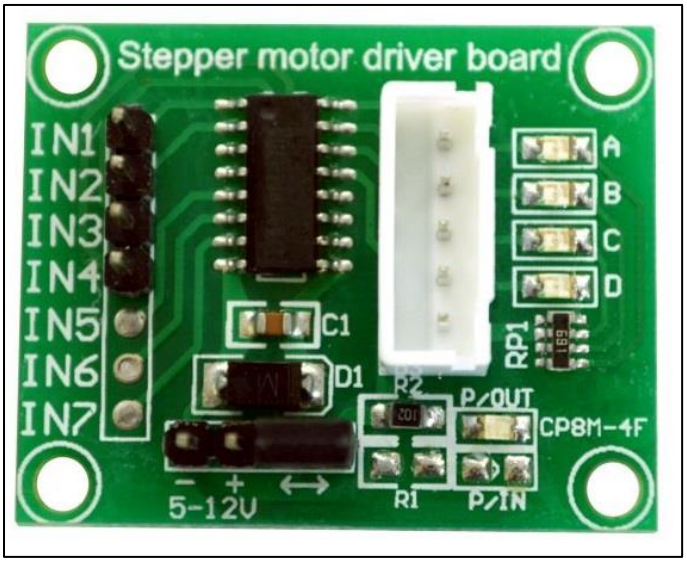

Figura 3.4. Driver para motor a pasos ULN2003A.

El sistema de iluminación está básicamente compuesto por una tira de led de 15 cm alimentada por un adaptador de corriente directa de 12V (Figura 3.5). Se optó por un sistema de iluminación continua debido a que el uso de flash mostro un problema visual de sombras.

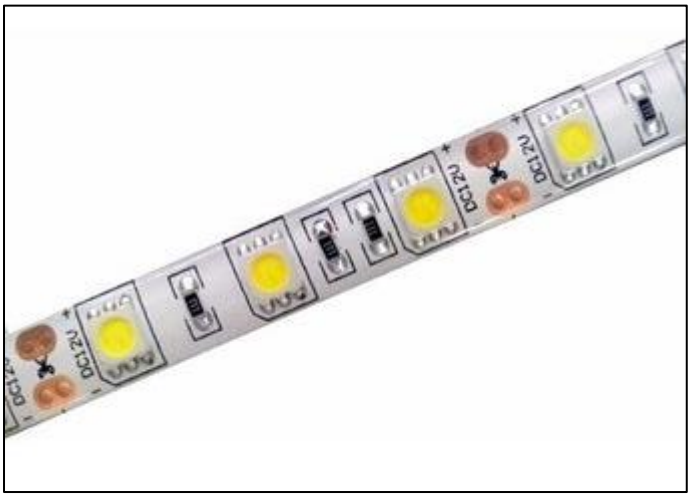

Figura 3.5. Tira de Led Blanca 3M.

La estructura general del prototipo está constituida por un cubo de triplay descubierto en su cara frontal desde la cual se lleva a cabo la captura de las fotografías del objeto a modelar, en su parte inferior cuenta con un pequeño compartimento encargado de resguardar la circuitería del prototipo (Figura 3.6).

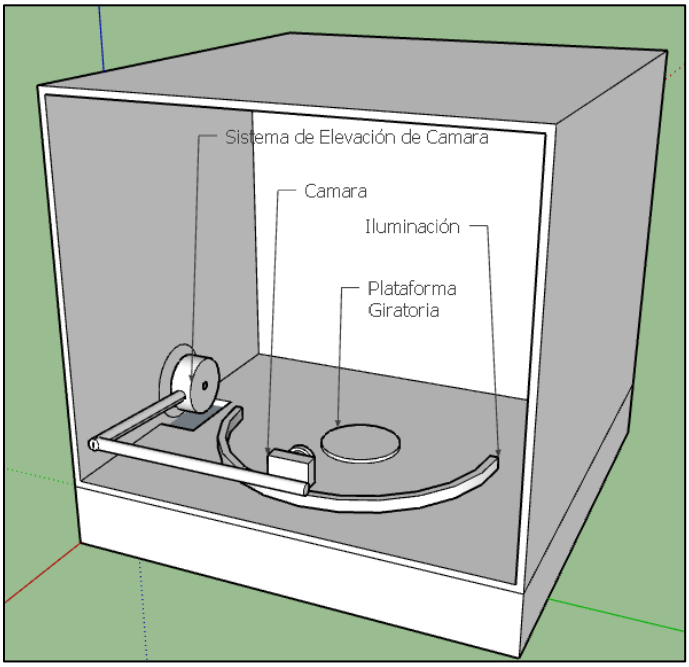

Figura 3.6. Estructura general del prototipo.

Fuente: Elaboración propia.

Por el lado del software se usaron esencialmente tres diferentes tipos. Inicialmente se utilizó Matlab (Figura 3.7) con la única finalidad de probar el funcionamiento de los módulos de reconstrucción tridimensional y calibración de cámaras para visión estéreo para poder entender los puntos básicos requeridos durante un proceso de reconstrucción común. Matlab es un poderoso paquete de software matemático y gráfico con capacidades numéricas, gráficas y de programación. Incluye un potente entorno de desarrollo basado en el simple uso de expresiones ingresadas por el usuario de manera secuencial a las cuales, siendo Matlab un intérprete, responde inmediatamente con un resultado. Además permite

generar scripts y programas que son esencialmente grupos de comandos ejecutados secuencialmente (Attaway, 2016).

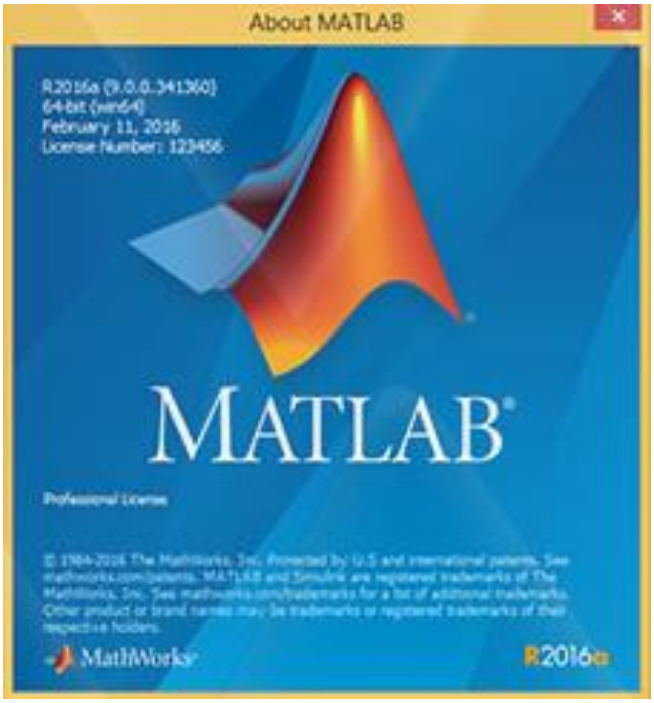

Figura 3.7. Pantalla inicial de Matlab R2016a.

Fuente: Elaboración propia.

A pesar de ser un software extremadamente útil para el tratamiento de imágenes, se decidió finalmente no utilizar Matlab debido a que cualquier cambio en el hardware de captura de imágenes conllevaba a un largo y complejo proceso de calibración del nuevo dispositivo instalado en el prototipo, mismo que debería repetirse cada vez que el usuario decidiera implementar una cámara con diferentes características a la anteriormente utilizada, dificultando el uso del prototipo para usuarios con conocimientos básicos de computo.

El segundo tipo de software utilizado fue Java, más en específico el entorno de desarrollo NetBeans. Este IDE funciona a manera de editor de texto, ayuda, compilador, depurador y cuenta con un amplio número de herramientas que pone a nuestra disposición para el diseño y programación de aplicaciones de escritorio. Netbeans puede ser adquirido de

manera gratuita en su web oficial (www.netbeans.org). Al ser un entorno de desarrollo de código abierto podemos encontrar en la web un gran número de complementos a manera de librerías que nos ayuden con el propósito de nuestra aplicación. La pantalla principal de desarrollo de NetBeans se muestra en la Figura 3.8:

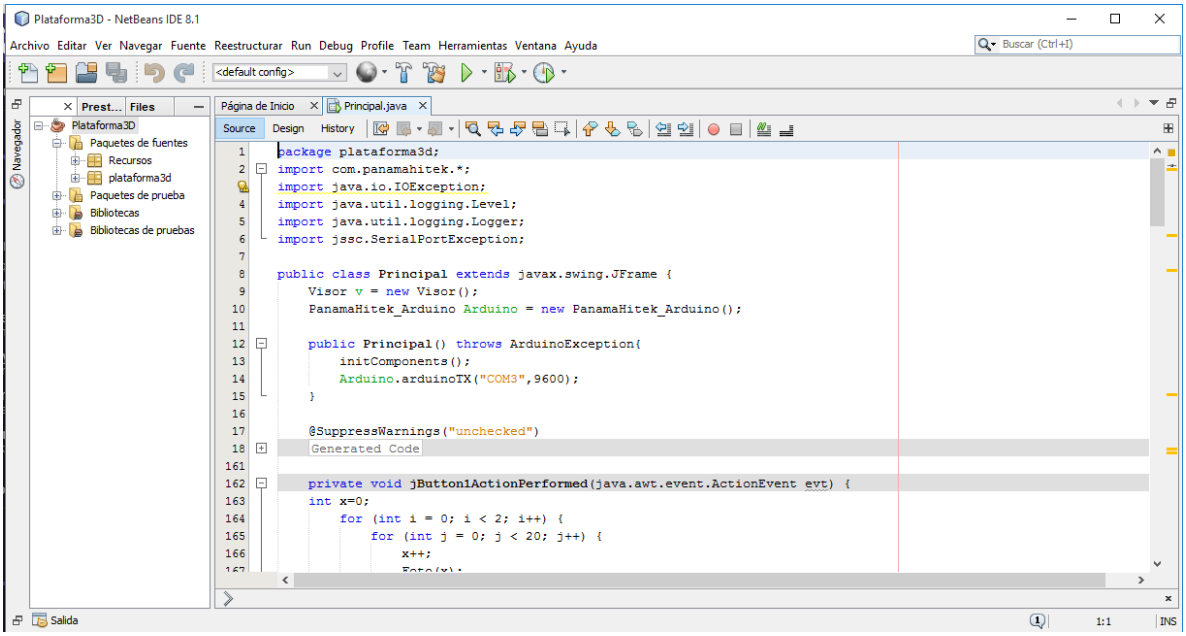

Figura 3.8. Interfaz principal de desarrollo de NetBeans.

Fuente: Elaboración propia.

El tercer software usado y pieza fundamental de este proyecto es producto de un proyecto de investigación llevado a cabo en 2015 por el Dr. Zhuoliang Kang (Kang, 2015). Este software llamado MeshRecon es una implementación de la plataforma de reconstrucción 3D del Ingeniero en Software Changchang Wu, de nombre VisualSFM. VisualSFM es básicamente una interfaz gráfica de usuario para reconstrucción 3D usando la técnica de estructura del movimiento (Wu, 2014), más adelante se hablara más a detalle de esta herramienta. Su interfaz principal se muestra en la Figura 3.9:

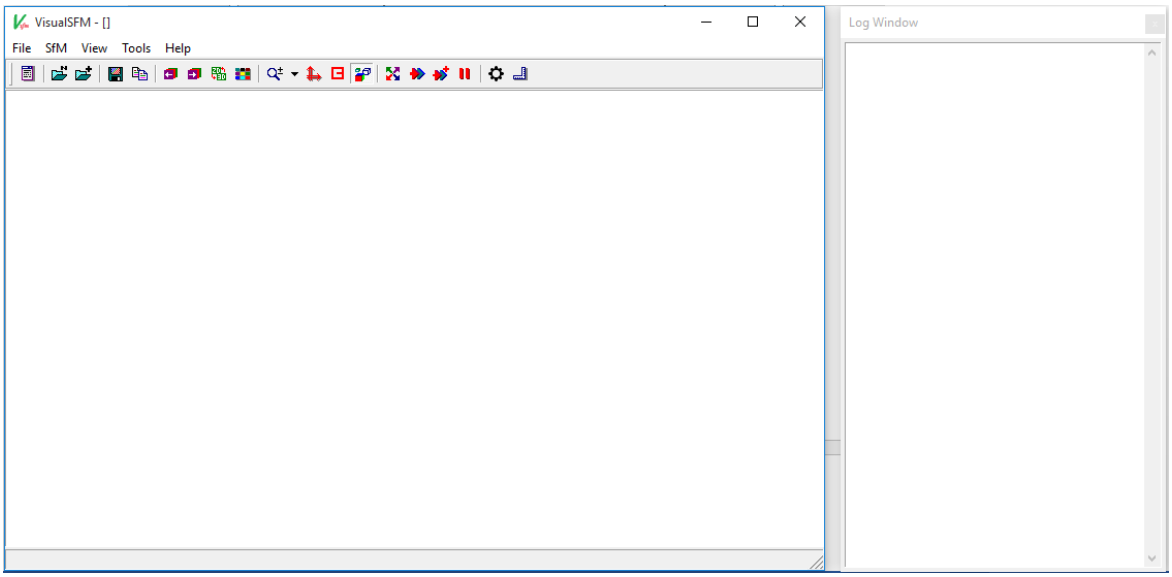

Figura 3.9. Pantalla principal de VisualSFM.

Fuente: Elaboración propia.

Metodología

Finalmente hay que mencionar que para llevar a cabo este proyecto se utilizó una metodología de desarrollo de hardware y software embebidos propuesta en (A. Perez, A. Ruiz de Olano, & J. Perez, 2006) y conocida como ciclo de desarrollo clásico V o metodología V. El concepto de embebido se refiere al hardware y software que trabajan en conjunto como un todo y de manera "invisible" al usuario.

En el modelo V se definen las siguientes etapas de desarrollo:

- Definición de especificaciones (Fase 1): Se deben definir y documentar los diferentes requisitos del sistema a desarrollar.
- Diseño global (Fase 2): También es llamado diseño de alto nivel. Su objetivo es obtener un diseño y visión general del sistema.
- Diseño en detalle (Fase 3): Consiste en detallar cada bloque obtenido de la fase anterior.
- Implementación (Fase 4): Es la fase en la que se materializa el diseño en detalle.

- Test Unitario (Fase 5): En esta fase se verifica cada módulo de hardware y software de forma unitaria, comprobando su funcionamiento adecuado.
- Integración (Fase 6): En esta fase se integran los distintos módulos que forman al sistema. Al igual que en la fase anterior se recomienda generar un documento de pruebas. Por una parte de comprobarse en todo el sistema el funcionamiento correcto, por otra, en caso de tratarse de un sistema tolerante a fallos, debe verificarse que ante la presencia de un fallo persiste el funcionamiento correcto. Se comprueba además el funcionamiento de los requisitos establecidos.
- Test operacional del sistema (Fase 7): Se realizan las últimas pruebas pero sobre un escenario real, en su ubicación final, registrando las pruebas llevadas a cabo y los resultados obtenidos.

En la Figura 3.10, se muestra de manera gráfica la estructura de la metodología V:

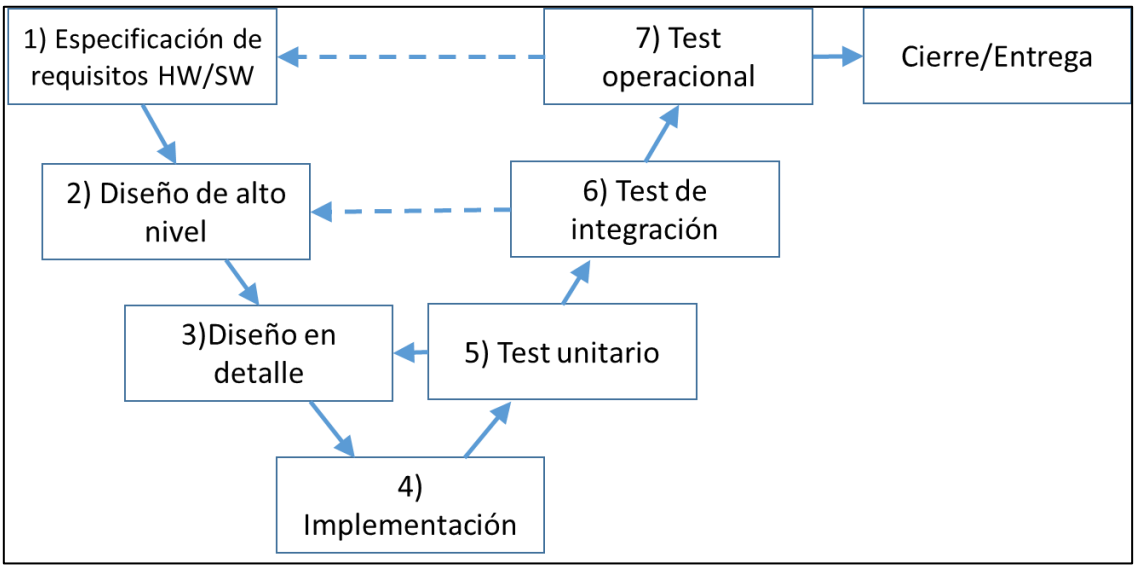

Figura 3.10. Diagrama general del ciclo de vida de la metodología V.

Fuente: (A. Perez, A. Ruiz de Olano, & J. Perez, 2006)

Capítulo 4 Trabajos relacionados / Tecnologías existentes

Durante la etapa de planeación de este proyecto un punto importante de investigación fue la búsqueda y recopilación de las tecnologías existentes y comúnmente utilizadas para llevar a cabo tareas de modelado 3D. A continuación se muestran algunos de los trabajos más relevantes encontrados durante esta etapa con la finalidad de analizar algunas de las técnicas existentes para modelar en 3D objetos del mundo real.

Modelado mediante el uso de escaneo laser

A lo largo del documento titulado "Construcción de un modelo digital 3D de piezas precolombinas utilizando escaneo laser" (Mesa Múnera, Ramírez Salazar, & Branch Bedoya, 2010) se propone una metodología para llevar a cabo el proceso de reconstrucción tridimensional de piezas pertenecientes al museo de la Universidad de Antioquia. Dicha metodología se compone de 7 etapas: Escaneo, Filtrado, Triangulación, Corrección, Registro, Integración y Ajuste.

Para la etapa de escaneo se seleccionaron piezas con variadas características como forma, reflectividad, tamaño, textura y complejidad y fueron sometidas a un escaneo mediante el uso del Escaner Zephyr Kreon 100 (Figura 4.1). Este escaner se compone básicamente de un brazo robótico con seis grados de libertad el cual proyecta de manera perpendicular un láser sobre la superficie del objeto obteniendo la posición de diferentes puntos del mismo en un espacio respecto a un sistema de coordenadas absoluto. Mediante el software Polygonia V15 se llevó a cabo el proceso de obtención de una nube de puntos con una alta densidad perteneciente a cada una de las caras del objeto que posteriormente fue procesada mediante un filtro encargado de eliminar datos redundantes en zonas de poca curvatura a fin de disminuir el costo computacional de procesamiento.

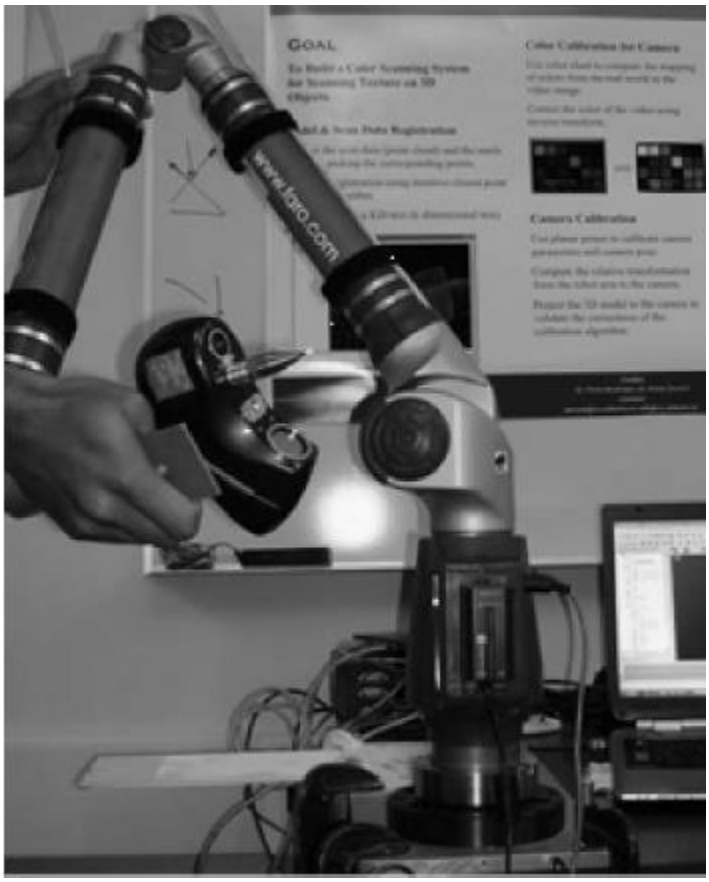

Figura 4.1. Escáner Zephyr Kreon 100.

Fuente: (Mesa Múnera, Ramírez Salazar, & Branch Bedoya, 2010)

Posteriormente se usa como referencia la nube de puntos generada inicialmente para llevar a cabo una malla triangular, es decir, uniendo cada uno de los puntos entre sí a manera de generar figuras triangulares contiguas hasta llenar por completo la nube. Durante la etapa de corrección se aplicó un algoritmo de identificación y llenado de agujeros en la malla. Dicho algoritmo define un "agujero" cuando identifica que al menos una de las aristas de un triángulo no está en contacto con otra. Estos huecos son llenados mediante el uso de funciones de base radial.

Finalmente se lleva a cabo el proceso de registro e integración del conjunto de mayas pertenecientes a cada cara del objeto mediante la detección de puntos coincidentes entre cada una de las caras a fin de unir cada una de las mayas para terminar con un modelo fiel

del objeto después de un ajuste de posición y un proceso final de corrección y rellenado de huecos.

En la Figura 4.2 se muestra el proceso general de reconstrucción propuesto a lo largo del documento citado:

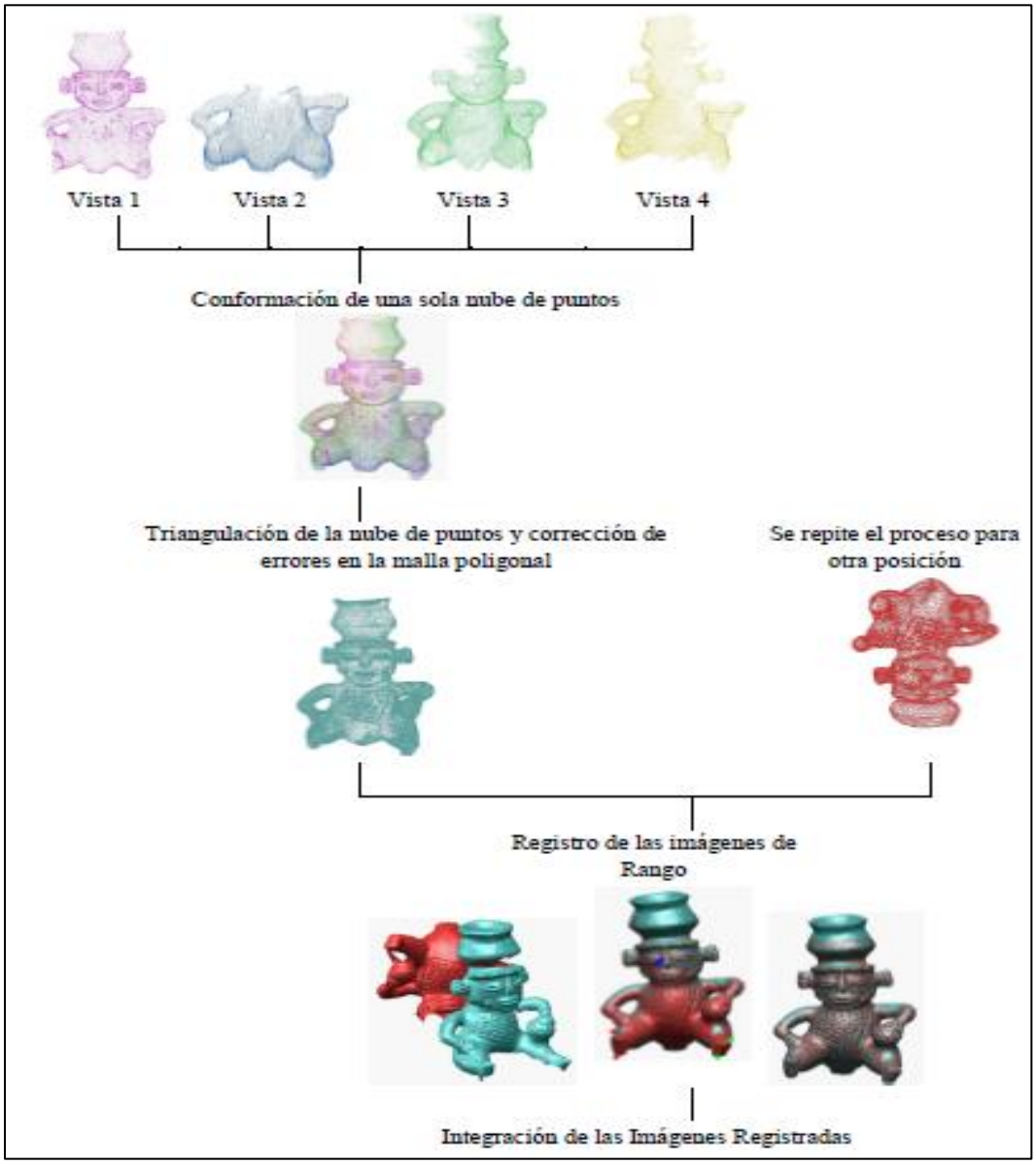

Figura 4.2. Esquema generalizado del proceso de reconstrucción.

Fuente: (Mesa Múnera, Ramírez Salazar, & Branch Bedoya, 2010)

Modelado mediante el uso de escaneo por sensor ultrasónico

Otra técnica comúnmente utilizada, tal como se describe en (Orozco Quinceno, Romero Acero, Marín Cano, & Jímenez Builes, 2014), es el uso de sensores ultrasónicos. A lo largo de este documento se plantea el diseño e implementación de un sistema de modelado y reconstrucción 3D compuesto por tres subsistemas: Electromecánico, sensores y computacional (Figura 4.3).

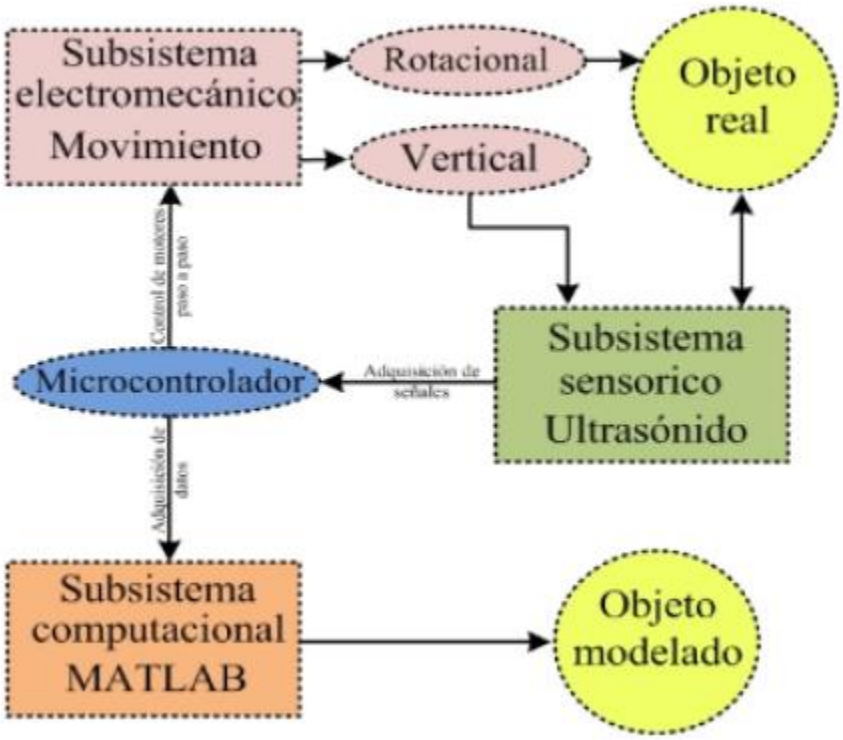

Figura 4.3. Estructura general del sistema de modelado.

Fuente: (Orozco Quinceno, Romero Acero, Marín Cano, & Jímenez Builes, 2014)

El subsistema electromecánico se encarga de llevar a cabo el movimiento vertical del sensor así como el movimiento rotacional del objeto a digitalizar. El sensor se mueve aproximadamente un centímetro con la finalidad de que a cada paso se tome la medición efectuada por el sensor, en cuanto al movimiento rotacional, éste se lleva a cabo en pasos de 4 grados dando como resultado un total de 90 pasos para completar una vuelta. Cabe destacar que para realizar dicho movimiento se utilizó un par de motores a pasos y una plataforma giratoria.

En cuanto al subsistema de sensores, éste consiste en la emisión cíclica de un impulso acústico de alta frecuencia y corta duración, dicho impulso se impacta con el objeto a modelar y es reflectado

de vuelta al sensor. En cada emisión se mide el tiempo que tarda en llegar el pulso desde su salida hasta su regreso al sensor, y tomando como referencia el centro de la base giratoria sobre la cual se coloca el objeto, se triangula la posición de éste en cada toma. Esta información es procesada y enviada por medio de un microcontrolador al subsistema de cómputo.

Finalmente el subsistema computacional está compuesto por un algoritmo escrito en Matlab que se encarga de triangular en un espacio dimensional cada registro tomado por el sensor y llevar a cabo un proceso de suavizado para eliminar datos atípicos finalizando con la reconstrucción del objeto mediante la triangulación Delaunay integrada en Matlab.

En la Figura 4.4 se muestra el prototipo empleado para llevar a cabo el escaneo.

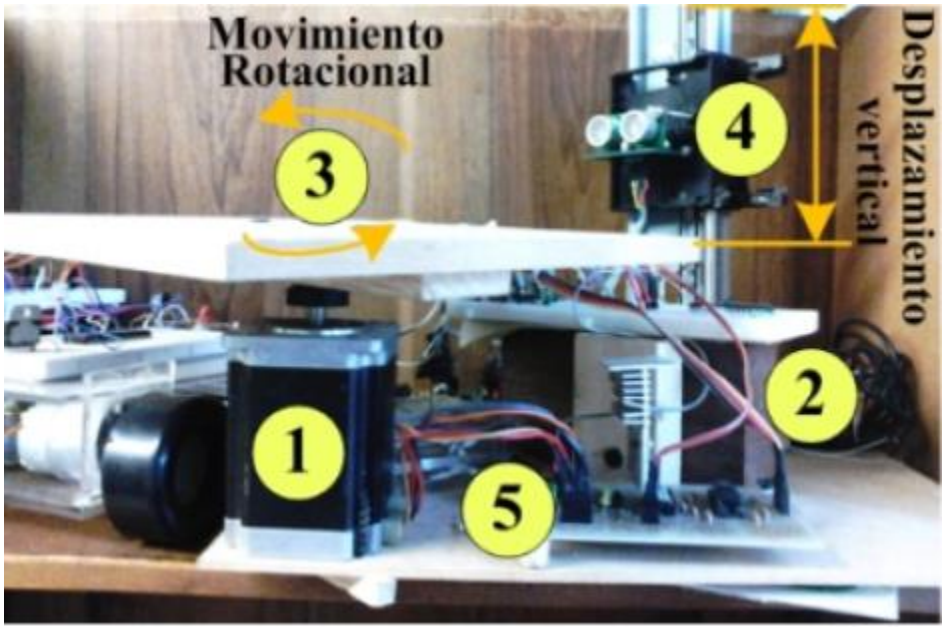

Figura 4.4. Prototipo final de escaneo. 1) Motor a pasos para movimiento rotacional. 2) Motor a pasos para movimiento vertical. 3) Plataforma giratoria. 4) Sistema sensorico. 5) Microcontrolador.

Fuente: (Orozco Quinceno, Romero Acero, Marín Cano, & Jímenez Builes, 2014)

Reconstrucción de modelos 3D mediante visión artificial

En el documento titulado “Reconstrucción de modelos 3D utilizando técnicas de visión artificial” (Vilá Ubieto, Arranz Domingo, Alvar Miró, & Sánchez Miralles, 2009) se plantea un sistema de reconstrucción por medio de visión multicámara, esto se refiere al uso de una o más fuentes de captura de imagen posicionadas en diferentes puntos de vista del objeto. Cabe destacar que este proyecto aplica únicamente con objetos de reconstrucción básica, es decir objetos con un gran número de aristas rectas y un mínimo número de curvaturas.

La metodología propuesta para este proyecto consiste en un inicio en la toma de imágenes del objeto desde una cámara estereoscópica Bumblebee2 colocada de manera paralela frente al objeto a modelar. Posteriormente se aplica un algoritmo en Matlab enfocado a la detección de esquinas del objeto. Estas esquinas serán posteriormente sometidas a un proceso de detección de bordes o aristas para definir la estructura básica del objeto y generar, a partir de ésta, una maya en tres dimensiones del mismo.

Finalmente se aplica un último algoritmo encargado de la búsqueda y detección de puntos de interés en las imágenes para llevar a cabo un ensamblaje de las mallas generadas por cada cámara.

Una desventaja del uso de esta técnica es que solamente se puede obtener la digitalización en tres dimensiones de una sola cara del objeto.

En la Figura 4.5 se muestran dos ejemplos de los resultados obtenidos a partir de la digitalización de un cubo Rubik y una engrapadora:

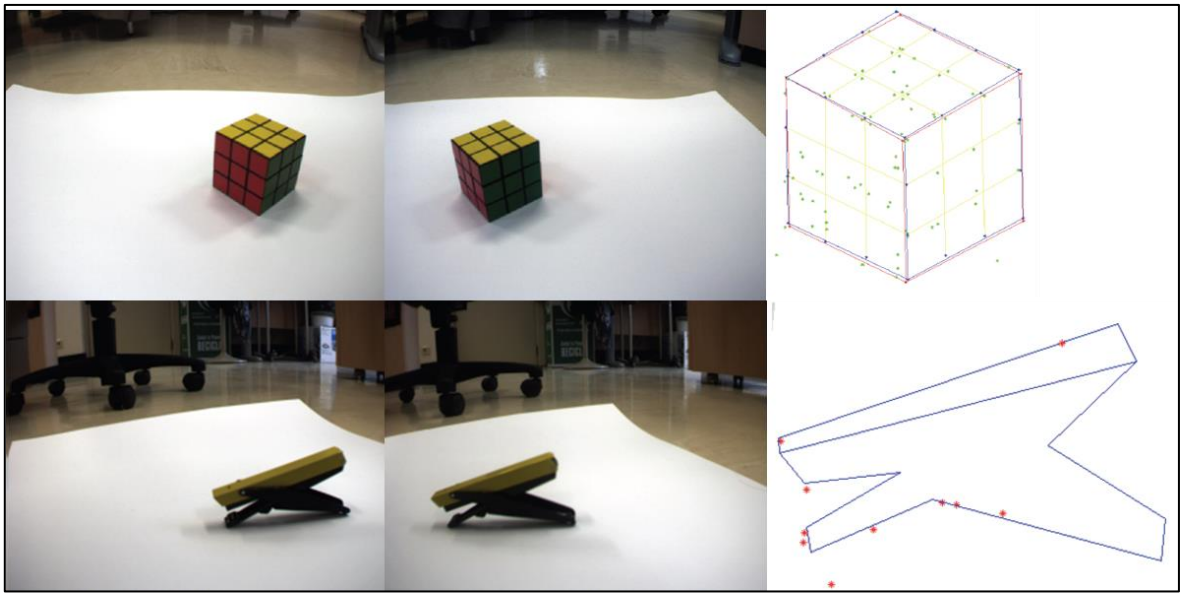

Figura 4.5. Modelos obtenidos a partir de un cubo Rubik y una engrapadora.

Fuente: (Vilá Ubieto, Arranz Domingo, Alvar Miró, & Sánchez Miralles, 2009).

Capítulo 5 El entorno de captura

Primer prototipo

Inicialmente, el módulo de Hardware se pensó como un simple escenario o estudio fotográfico a baja escala con la única finalidad de facilitar la captura de imágenes de un objeto real. Este primer prototipo constaba únicamente de un área de captura en forma de medio cilindro en color blanco dentro del cual se colocaría el objeto a modelar, al centro de este una plataforma giratoria de forma circular sobre la cual el objeto al ser colocado pudiera girarse de forma manual sobre su propio eje con la finalidad de permitir la captura desde diferentes puntos de vista del mismo. Finalmente, en la parte frontal del prototipo se encontraría un soporte o zona específica en la cual se colocaría el dispositivo de captura de imágenes (Telefono celular, Camara digital, Camara Web, etc.). Este prototipo se diseñó primero de manera digital tal y como se muestra en la Figura 5.1:

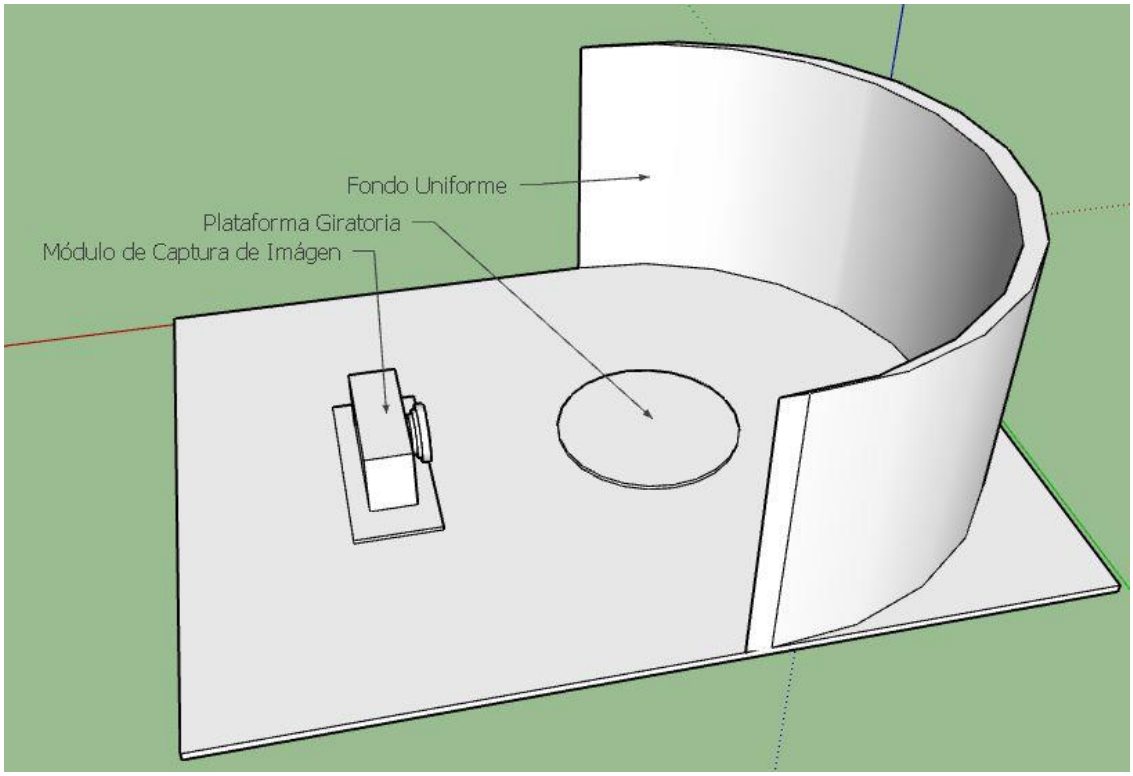

Figura 5.1. Modelo digital del primer prototipo del módulo de hardware.

Fuente: Elaboración propia.

Una vez lista la primera propuesta de manera digital, ésta fue plasmada de manera física en cartón con la única finalidad de probar la eficiencia de la misma al momento de capturar las imágenes de algún objeto así como detectar las deficiencias de la misma. El resultado se muestra en la Figura 5.2 y Figura 5.3:

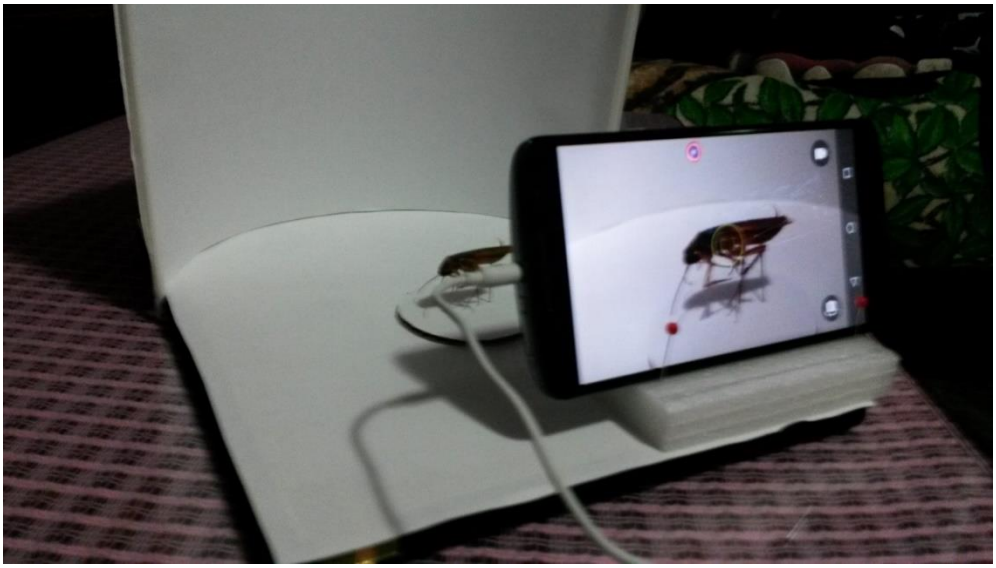

Figura 5.2. Primer prototipo del módulo de captura de imagen.

Fuente: Elaboración propia.

Figura 5.3. Primer prototipo del módulo de captura de imagen.

Fuente: Elaboración propia.

Una vez iniciadas las pruebas de adquisición de imágenes llevadas a cabo mediante un teléfono celular, las principales deficiencias de dicho prototipo se hicieron obvias al instante. Como se muestra en la Figura 5.4, las fotografías capturadas mediante este método presentan un exceso de sombras causadas por el disparo del flash del celular, mismo que sería ocasionado con el flash de una cámara digital. Por otro lado, la falta de iluminación dificulta también el reconocimiento del objeto si la luz no es suficiente para la captura. Además, como se alcanza a percibir en las fotografías de ejemplo, se percibe una línea curva al fondo de la imagen producto de la intersección entre la pared semi curva y la base del escenario. Todos estos elementos se traducen como ruido dentro de la imagen, computacionalmente hablando esto supone un mayor tiempo de procesamiento, una menor eficiencia y por consecuencia modelos que presentaran deformidades e inexactitudes.

Figura 5.4. Imágenes obtenidas de un insecto mediante el primer prototipo de captura.

Fuente: Elaboración propia.

Debido a todos estos detalles en conjunto fue que se decidió optar por generar un nuevo prototipo que subsanara dichas problemáticas.

Segundo Prototipo

La segunda propuesta al módulo de hardware tenía como objetivo subsanar en gran medida los problemas detectados durante la etapa de planeación y prueba del primer prototipo. Para ello se reemplazó la estructura cilíndrica por una estructura externa en forma cubica constituida por paredes de cartón duro en color blanco. Además se incorporó un fondo curvo, este fondo tiene la finalidad de eliminar en mayor medida de lo posible el "ruido" en las imágenes, es decir manchas, líneas y sombras, en mayor medida de lo posible. Este fondo curvo permite mantener un entorno de captura controlado e implementa un efecto óptico conocido como "fondo infinito", esta técnica se usa regularmente en la fotografía de productos en el campo de la mercadotecnia y permite simular un fondo sin fin detrás del objeto en cada toma (Guasco, 2012).

En cuanto al sistema de iluminación, se implementaron dos tiras de Led de alta luminosidad de 30 cm colocadas en paralelo para suministrar una fuente de luz constante. Al igual que con el prototipo anterior, éste está formado por una plataforma giratoria manual que permite al objeto realizar un movimiento rotatorio sobre su eje vertical y una abertura frontal desde la cual se colocaría el dispositivo de captura.

En la Figura 5.5 se muestra el segundo prototipo ya montado y en funcionamiento:

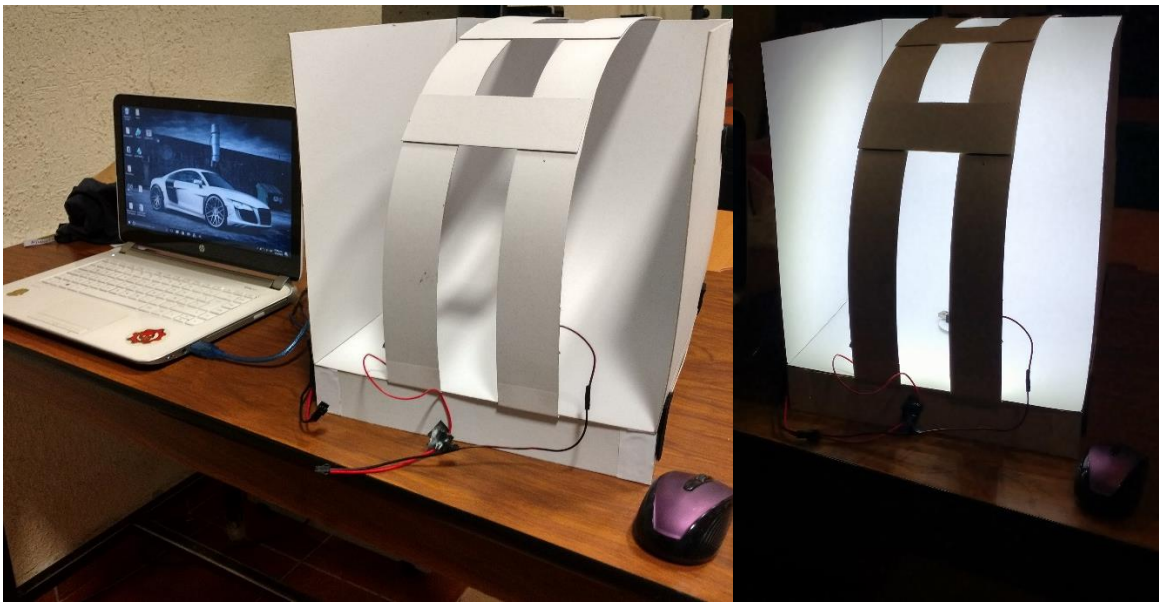

Figura 5.5. Segundo prototipo de captura.

Fuente: Elaboración propia.

Una vez iniciadas las pruebas del proceso de captura, las deficiencias del nuevo prototipo surgieron nuevamente. La principal fue el sistema de iluminación. A pesar de haber logrado eliminar casi en su totalidad las sombras del objeto en su entorno, la tira del led generaba una distorsión en forma de líneas negras sobre la toma (ver Figura 5.6) debido a la incompatibilidad entre la frecuencia de iluminación y la tasa de refresco de la cámara.

Figura 5.6. Distorsión de líneas generada por la frecuencia de iluminación.

Fuente: Elaboración propia.

La frecuencia (o tasa) de refresco de una cámara se refiere a cuantas imágenes puede capturar la cámara por segundo, y su unidad de medida son los Frames Per Second (FPS) que, relacionándolo a la frecuencia de iluminación, se traduce en una incompatibilidad con los Hz (Hercios). Por lo tanto, si la diferencia entre ambos es muy grande, se genera esta distorsión en forma de líneas debido a la nula sincronización entre los destellos de luz y el índice de refresco de la cámara (Vázquez, 2016).

Prototipo final

Para el diseño y desarrollo del prototipo final se dio prioridad a 3 puntos esenciales:

- La estructura final
- El sistema de iluminación
- Portabilidad

La estructura del prototipo anterior demostró ser eficiente y mantener el entorno de captura de manera controlada por lo cual se decidió mantenerse el mismo con un par de modificaciones adicionales. La primera de éstas fue cerrar por completo la estructura a excepción de la cara frontal de la misma, esto con la finalidad de evitar en mayor medida de lo posible el ingreso de luz que pudiera perjudicar al proceso de captura de las imágenes. Además, dicha estructura fue cambiada de cartón a madera tipo Triplay en tono blanco mate para evitar el exceso de reflejo de luz tanto exterior como del mismo sistema de iluminación. En la parte inferior del prototipo se integró un compartimento especial dentro del cual se resguarda la mayor parte de la circuitería del proyecto.

En cuanto al sistema de iluminación, éste se sometió a diferentes pruebas con tiras de led de diferentes tipos, desde tiras de led sencillas, tiras de led híper led (máxima luminosidad) y hasta la implementación de leds de potencia con una fuente de alimentación de 12 V (ver Figura 5.7).

Figura 5.7. Leds de potencia y fuente de alimentación de 12V con capacidad para 9 leds.

Fuente: Elaboración propia.

Finalmente se optó por la implementación de una tira del led de frecuencia baja de la marca 3M a 20 Hz con la cual se logró subsanar el problema de distorsión de líneas. Esta tira de led de 30 cm fue montada sobre un soporte curvo para optimizar el direccionamiento de la luz sobre el objeto a modelar (Figura 5.8).

Figura 5.8. Tira de led 3M de 12V.

Fuente: Elaboración propia.

La estructura final propuesta fue diseñada inicialmente de manera digital tal y como se muestra en la Figura 5.9.

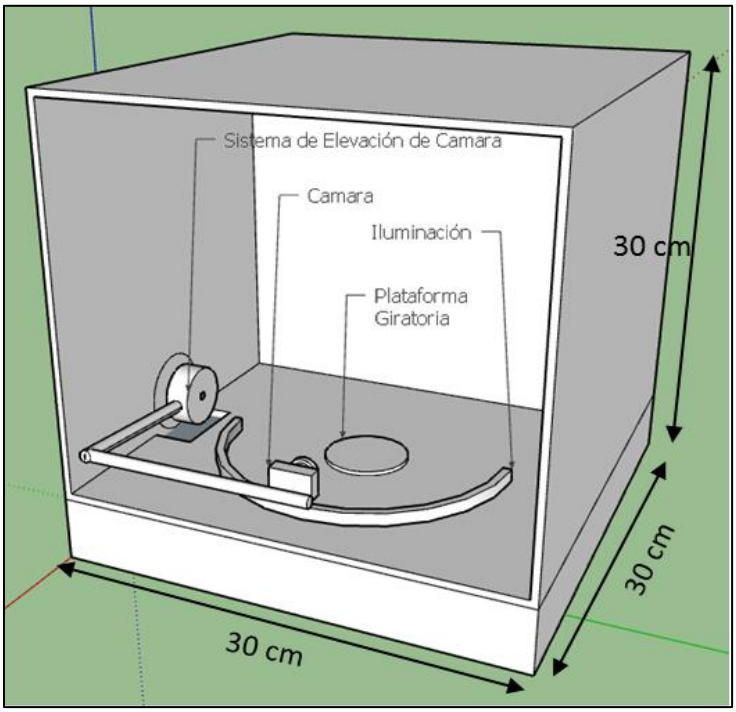

Figura 5.9. Boceto de la estructura final del dispositivo de modelado.

Fuente: Elaboración propia.

Uno de los puntos más representativos del prototipo fue la portabilidad. Esto es, se diseñó la estructura del dispositivo de manera que ésta pueda ser montada y desmontada por completo de manera sencilla al usuario, protegiendo siempre la circuitería antes mencionada, con la finalidad de facilitar al usuario el transporte y resguardo del aparato.

Dicha estructura se muestra en la Figura 5.10:

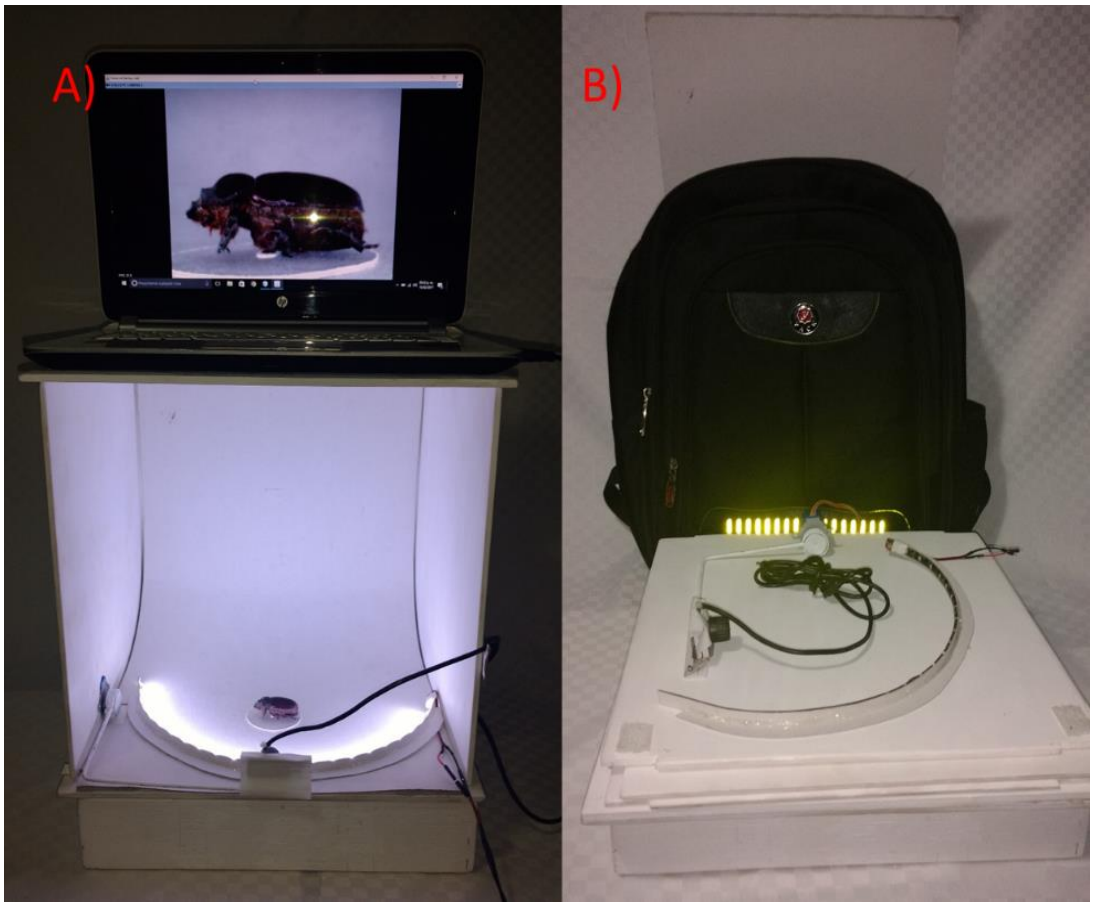

Figura 5.10. A) Estructura Final en funcionamiento B) Estructura final completamente desmantelada y lista para transportar en una maleta.

Fuente: Elaboración propia.

Capítulo 6 Módulo de Hardware

Alternativas a Arduino

Como se mencionó con anterioridad, actualmente existen muchas alternativas de hardware libre en el mercado que nos ofrecen una funcionalidad similar a la de las placas Arduino.

Raspberry Pi es un mini ordenador incorporado en una placa reducida de 85*56 mm con todos los componentes necesarios para poder ejecutar en ella un sistema operativo. Fue diseñada y desarrollada por la fundación Respberry Pi con la intención de poner a disposición de niños y adolescentes un ordenador a bajo costo que les permitiera descubrir y adentrarse en el mundo de la informática y la programación (MOCQ, 2016). Las características de este dispositivo varían según el modelo, por ejemplo, el modelo Raspberry Pi B+ incorpora entre sus componentes un circuito integrado Broadcom BCM2835 a 700MHz con 512 MB de RAM, soporte para 40 pines GPIO (General Purpose Input/Output), Salida de video tipo HDMI, 4 puertos USB y un slot de memoria tipo Micro SD (Pi, 2017). Este modelo de Raspberry Pi se muestra en la Figura 6.1 y tiene un costo aproximado de $600.0 M/N:

Figura 6.1. Raspberry Pi B+.

Por otro lado, BeagleBone es el resultado del esfuerzo de un grupo de personas interesadas en crear herramientas potentes e integradas de código abierto. Se trata de mini-computadoras de placa única sin ventilador y a bajo costo basados en procesadores Texas Instruments de baja potencia con núcleo de la serie ARM Cortex-A con soporte para las plataformas Linux, Ubuntu, QNX, Windows Embedded, herramientas web e incluso programación para Arduino. Al igual que con Raspberry las características varían según el modelo, por ejemplo, la versión Pocket Beagle (Figura 6.2) está compuesta por un procesador Octavo Systems ARM Cortex-A8 de 1GHz, 512 MB de memoria RAM DDR3, puerto microUSB y un cabezal de 72 pines GPIO (Kridner & Long, 2017). Este modelo tiene un precio aproximado de $800.0 M/N.

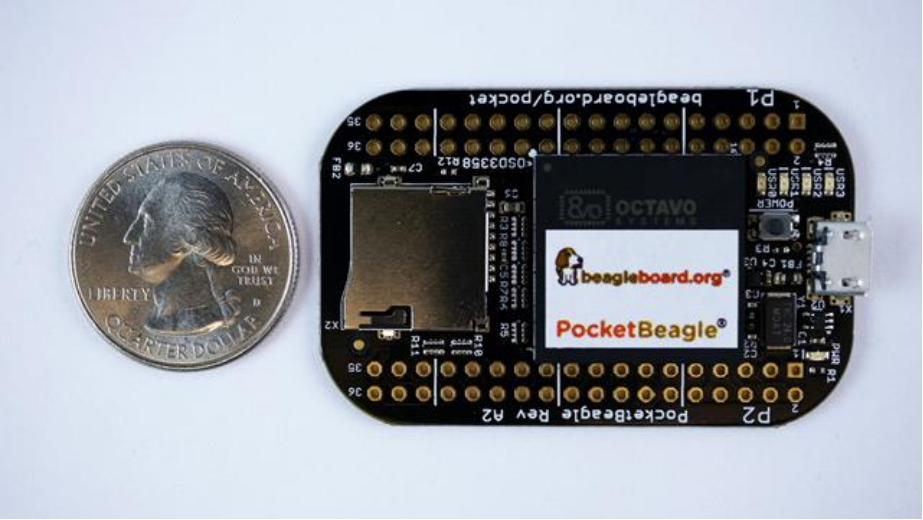

Figura 6.2 TarjetaPocket Beagle.

¿Por qué Arduino?

Sin embargo, hay algunas características propias de Arduino que lo hacen tener un buen número de ventajas significativas respecto a sus contrincantes en el mercado, sobre todo en el campo del desarrollo tecnológico y de investigación. Entre las más relevantes podemos mencionar:

- Costo: Las placas Arduino son relativamente más baratas que otras alternativas. Tomemos como ejemplo uno de los dispositivos más económicos de hardware pertenecientes a la marcara Rasphberry, el Raspherry Pi Zero, tiene un costo

aproximado de $600 M/N, en contraposición podemos encontrar una de las versiones más económicas de la placa Arduino, el modelo Arduino Mini 05 Atmega168 tiene un costo regular de aproximadamente $300 M/N. Además de esto debemos considerar que al ser hardware libre su diagrama de ensamblaje se puede encontrar en la página oficial de Arduino por lo que la versión más económica de este puede ser ensamblada a mano por el usuario.

- Comunidad Arduino: Arduino es distribuido bajo una licencia Creative Commons Attribution Share Alike, lo cual significa que existe una gran comunidad de usuarios a nivel mundial usando el diseño de Arduino, modificándolo y compartiendo con el resto de usuarios sus evoluciones. Gracias a esto, miles de usuarios pueden consultar dudas y proponer ideas para mejorar el diseño de esta herramienta (Arduino, 2017).
- Es multiplataforma: A diferencia de la mayoría de sus competidores, que se encuentran limitados a Windows, el software de Arduino puede ser ejecutado en los sistemas operativos Windows, Macintosh y GNU/Linux (Arduino, 2017).
- Entorno de desarrollo simple y amigable: Como se mencionó en la sección Materiales y Métodos, el editor de Arduino está basado en Procesing y cuanta con un IDE amigable con el usuario que facilita su uso tanto para expertos en programación como para usuarios que comienzan a adentrarse en este mundo.
- Código abierto y software extensible: El software Arduino está publicado bajo licencia de código abierto, además de que como se menciona en la página oficial de Arduino, la placa Arduino puede ser programada bajo funciones en C/C++ e incluso cabe la posibilidad de usar IDE's independientes a Arduino tales como Makefiles o AVR Studio. Incluso se ofrece una plataforma en línea para programar la placa desde el explorador de la computadora sin tener que instalar el IDE de Arduino (Arduino, 2017).
- Variedad: Arduino cuenta con diferentes versiones que se diferencian entre si según las características técnicas y prestaciones de cada una. Con una variedad de más de

30 modelos diferentes fácilmente podemos encontrar en el mercado al menos una que se ajuste a las necesidades de nuestro proyecto.

- Características: Para el desarrollo de este prototipo las características correspondientes a Arduino son más que suficientes para cumplir con el trabajo, por lo cual el uso de un dispositivo más avanzado seria innecesario.

El proceso de automatización

Para llevar a cabo la tarea de automatización del sistema de movimiento fue necesaria la implementación de dos motores a pasos modelo 28BYJ-48. El primero de ellos se colocó internamente al compartimento inferior del prototipo, y se encarga de girar el objeto en un total de 20 pasos, esto con la finalidad de permitir capturar fotografías desde 20 diferentes puntos de vista del objeto tomando como referencia su eje vertical. El segundo motor fue posicionado en la parte lateral izquierda de la estructura del dispositivo, a este último le fue fijado un "brazo" metálico en ángulo de 90° que a su vez tiene montado el dispositivo de captura de imagen, una cámara USB de 8MPx de la marca Ele-Gate. Este segundo motor tiene la función de colocar la cámara en tres posiciones respecto al objeto para permitir la captura desde diferentes ángulos del mismo en cada paso del motor 1.

Ambos motores deben estar correctamente sincronizados para lograr un correcto conjunto de tomas mediante la placa Arduino Mega 2560. Para esto se utilizaron los pines 2-9 para la conexión y control de los motores y la salida de 5V y GND de Arduino para la alimentación de sus respectivos drivers. El esquema de conexión de la placa Arduino se muestra en la Figura 6.3, mientras que el funcionamiento general del circuito electrónico se describe en el diagrama de bloques de la Figura 6.4, en este se puede ver que el equipo de cómputo se encarga de sincronizar el movimiento de los motores a través de Arduino, así como de la captura de fotografías mediante la cámara web, mientras que el sistema de iluminación es autónomo y constante durante todo el proceso:

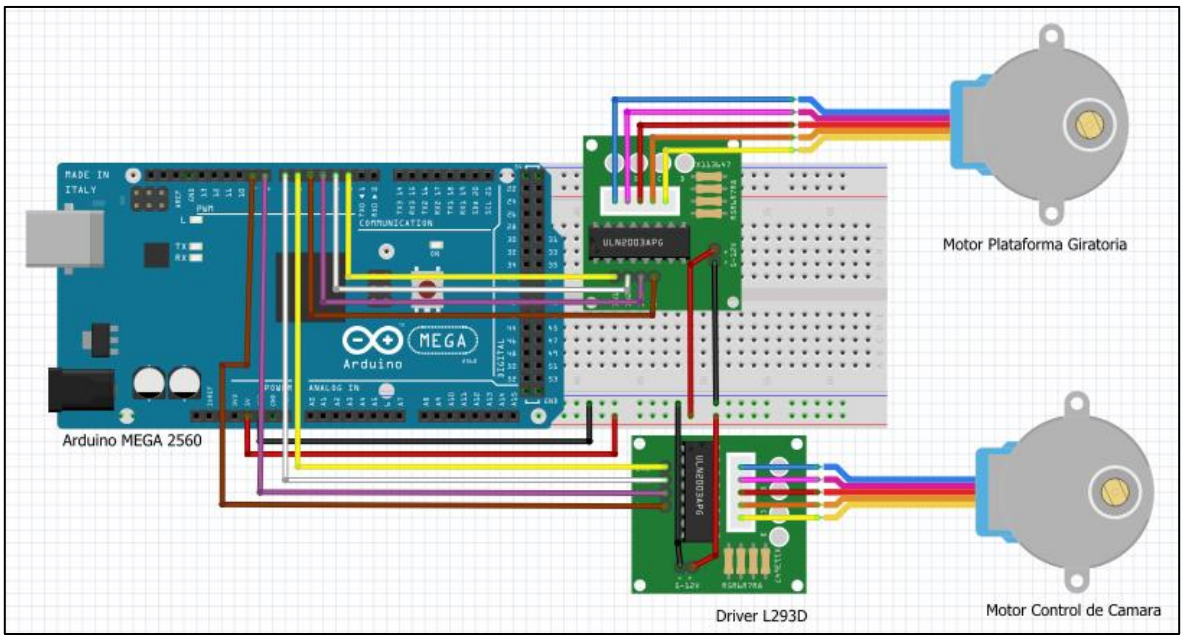

Figura 6.3. Esquema de conexión físico.

Fuente: Elaboración propia.

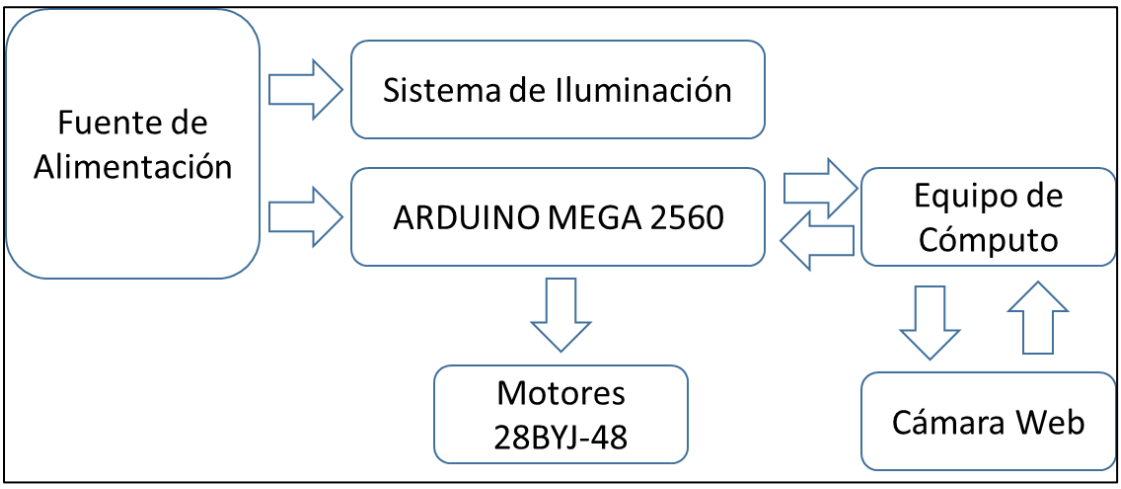

Figura 6.4. Funcionamiento general del circuito eléctrico.

Fuente: Elaboración propia.

Capítulo 7 Módulo de Software

Para desarrollar el módulo de software se trabajaron cinco elementos por separado que en conjunto integran un sistema mayormente generado en Java que permite llevar a cabo la calibración de la cámara en tiempo real y la captura del conjunto de imágenes final de manera automática mediante el control sincronizado de ambos motores y la comunicación directa con la cámara.

Finalmente se incrusto dentro de este software un script de modelado conocido como MeshRecon que permite, de manera nativa, llevar a cabo la tarea de digitalización con las imágenes obtenidas en el paso anterior.

Algoritmo general

El flujo de procesos llevados a cabo para capturar la totalidad de imágenes requeridas para el modelado es el siguiente:

1) Inicia la ejecución del programa.
2) Llevar a cabo la calibración manual de la cámara (Enfoque y ubicación).
3) Etiquetar ejemplar.
4) Se establece el inicio de la primera sesión de 3 en un ángulo de 0°.
5) Se establece el inicio de la primer toma de 20 (1/20).
6) Se captura la imagen.
7) Si la imagen capturada es la numero 20/20 se avanza al proceso 10, de lo contrario seguimos al proceso 8.
8) Se establece la captura de la imagen siguiente del conjunto de 20.
9) El objeto se gira un ángulo de 18° y se continúa al proceso 6.
10) Si la sesión actual es la 3/3 se finaliza el proceso de captura, de lo contrario se pasa al proceso 11.
11) Se establece una nueva sesión de capturas.
12) Se eleva la cámara en un ángulo de 20° respecto a su posición actual y se continúa en el proceso número 5.

Una vez terminado el proceso de captura, el software continua con el procesamiento de las imágenes obtenidas para que a partir de éstas, generar un modelo 3D del objeto en cuestión. (Ver Figura 7.1)

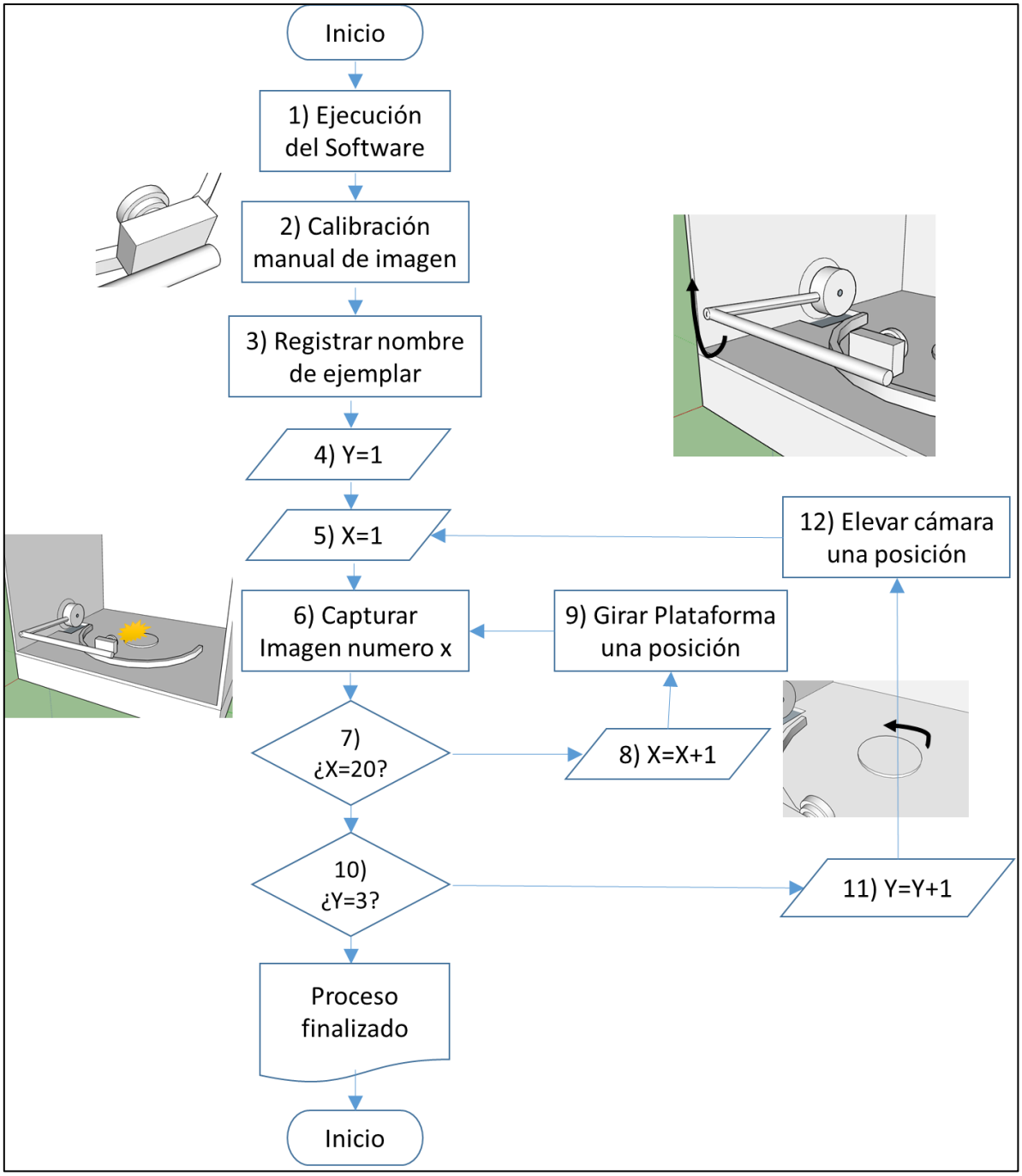

Figura 7.1. Diagrama de flujo general del proyecto.

Fuente: Elaboración propia.

Script Arduino

La programación de la tableta Arduino se lleva a cabo mediante la codificación de un script con extensión .ino el cual debe estar en una carpeta o directorio con el mismo nombre del script.

La estructura básica de un programa o script de Arduino consta de dos partes esenciales, la primera se refiere a la inclusión de bibliotecas y la declaración de variables y métodos, esta sección es identificada mediante la función Setup() y es ejecutada una sola vez, además tiene la función de configurar el "pinMode", es decir, determinar si un pin digital será usado como entrada o salida de datos e inicializar la comunicación en serie. Posteriormente, en la segunda sección identificada mediante la función loop(), se incluye el código que será ejecutado continuamente en un bucle infinito hasta terminar con la ejecución del programa, además de verificar en cada repetición el estado de los pines y llevar a cabo las instrucciones definidas por el usuario.

A continuación, en el Ejemplo 1 se muestra el código básico para activar y desactivar un pin digital, este puede ser usado, por ejemplo, para encender y apagar un led de manera intermitente.

```
Int pin = 10;
void setup() {
   pinMode(pin, OUTPUT);    // Establece 'pin' como salida
}
void loop() {
   digitalWrite(pin, HIGH); // Activa 'pin'
   delay(1000);             // Pausa un segundo
   digitalWrite(pin, LOW);  // Desactiva 'pin'
   delay(1000);
}
```

Ejemplo 1. Código básico para activar y desactivar un pin.

Fuente: Elaboración propia.

La correcta ejecución del script Arduino depende de su correcta configuración respecto a la placa que se está utilizando, dicha configuración se lleva a cabo en la interfaz principal de Arduino mediante el menú Herramientas, en el cual se selecciona en el campo "Placa" el modelo en que se desea cargar el código (Figura 7.2).

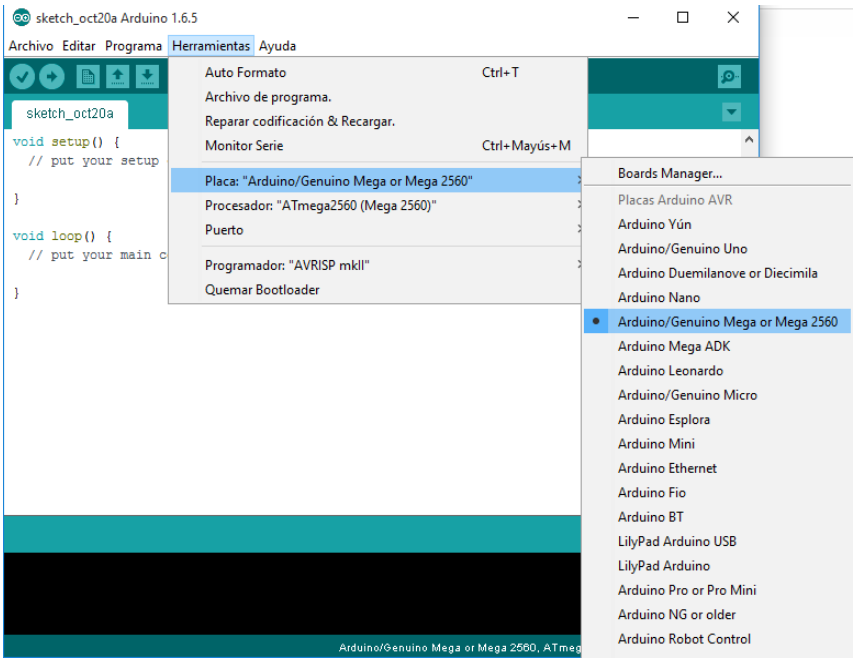

Figura 7.2 Configuración de la Tableta Arduino en el IDE oficial.

Fuente: Elaboración propia.

En cuanto al uso de librerías, únicamente se utilizó la librería Stepper.h, misma que se encuentra incluida en el IDE de Arduino y sirve como puente para llevar a cabo el control de motores a pasos.

Además se utilizaron un total de 9 variables, una de tipo char que recibe un dato desde la computadora la cual indica que acción deben realizar los motores (Mover la base, mover el brazo mecánico o reincorporar el brazo a su estado inicial) y 5 variables de tipo int que representan cada uno de los polos, cuatro por cada motor, sobre los cuales debe llevarse a cabo la excitación en el motor para que éste dé un paso a la izquierda o a la derecha y una última variable que funciona como contador.

En el Ejemplo 2 se muestra la configuración de pines utilizada para el control de los motores, como se puede ver se utilizó para esto los pines 2 a 9 que serían conectados de manera física a los polos de excitación del motor.

```
pinMode(5, OUTPUT);   // Pin 5 conectar a IN4
pinMode(4, OUTPUT);   // Pin 4 conectar a IN3
pinMode(3, OUTPUT);    // Pin 3 conectar a IN2
pinMode(2, OUTPUT);    // Pin 2 conectar a IN1

pinMode(9, OUTPUT);   // Pin 9 conectar a IN4
pinMode(8, OUTPUT);   // Pin 8 conectar a IN3
pinMode(7, OUTPUT);    // Pin 7 conectar a IN2
pinMode(6, OUTPUT);    // Pin 6 conectar a IN1
```

Ejemplo 2. Configuración de pines en la tableta Arduino.

. Fuente: Elaboración propia.

Dentro de la función loop() la codificación inicia con una sentencia if encargada de verificar que haya datos en espera de ser leídos a través del puerto de comunicación serial, es decir desde la computadora, de ser así el dato es leído y almacenado en la variable ”dato” y posteriormente mediante la sentencia switch se somete a una serie de decisiones para verificar si el dato de entrada es una solicitud de movimiento para la base giratoria o para el brazo de la cámara. Dicho movimiento es llevado a cabo mediante una instrucción for para cada caso encargado de repetir los métodos de un paso a la izquierda, un paso a la derecha o apagado del motor y continua con una breve pausa de tres segundos que es usada para capturar y guardar la fotografía del objeto en cada paso. Cada bucle finaliza con el método de apagado del motor con la intención de evitar el sobrecalentamiento del mismo.

Dicho código se muestra a continuación en el Ejemplo 3.

```
void loop() {

 //20 saltos para una vuelta de la base giratoria

 if(Serial.available()){
  dato=Serial.read();
  delay(3);
  switch(dato){
    //515 Pasos = una vuelta
   case '1': //Giro Base
    for(j=1;j<=26;j++){
        paso_der(2,3,4,5);
        apagado(2,3,4,5);
       }
       delay(300);
    break;

   case '2': //Giro Cámara
    for(l=1;l<=45;l++){
       paso_der(6,7,8,9);
       apagado(6,7,8,9);
      }
    break;

   case '3': //Retorno Cámara
    for(l=1;l<=90;l++){
       paso_izq(6,7,8,9);
       apagado(6,7,8,9);
      }
      delay(300);
    break;
  }
 }
 apagado(2,3,4,5);       // Apagado del Motor para evitar sobrecalentamiento
}
```

Ejemplo 3. Función loop() para el control de los motores.

Fuente: Elaboración propia.

Finalmente las funciones paso reciben como parámetro los pines inicialmente configurados para cada motor y se encargan de excitar cada polo secuencialmente hasta completar una vuelta completa del eje del mismo con la finalidad de dar un paso en la dirección deseada. En el Ejemplo 4 se muestra la codificación de uno de estos métodos.

```
void paso_der(int a,int b,int c,int d){        // Pasos a la derecha

 digitalWrite(d, LOW);
 digitalWrite(c, LOW);
 digitalWrite(b, HIGH);
 digitalWrite(a, HIGH);
  delay(3);
 digitalWrite(d, LOW);
 digitalWrite(c, HIGH);
 digitalWrite(b, HIGH);
 digitalWrite(a, LOW);
  delay(3);
 digitalWrite(d, HIGH);
 digitalWrite(c, HIGH);
 digitalWrite(b, LOW);
 digitalWrite(a, LOW);
  delay(3);
 digitalWrite(d, HIGH);
 digitalWrite(c, LOW);
 digitalWrite(b, LOW);
 digitalWrite(a, HIGH);
  delay(3);

}
```

Ejemplo 4. Codificación de la función paso derecha.

Fuente: Elaboración propia.

Para el caso del método de apagar motor todos los pines deben ser colocados en un estado LOW.

Conexión de Arduino con Java

Al igual que con Arduino, Java también cuenta con un amplio abanico de librerías que pueden ser cargadas al IDE con la finalidad de integrar nuevas funciones a los programas que se están codificando en este lenguaje, para este caso se utilizó la librería PanamaHitek_Arduino (García González, 2016) la cual está diseñada para simplificar la conexión de una aplicación de Java a la

Plataforma Arduino mediante los drivers para Windows rxtxSerial.dll. Estos drivers se encargan de establecer un medio de comunicación serial vía puerto USB entre la computadora y el puerto de alimentación de Arduino.

Para agregar esta librería se da clic derecho sobre el folder Bibliotecas que se encuentra en el panel lateral izquierdo de la interfaz principal dentro de la pestaña Projects y se selecciona la opción Agregar biblioteca, posteriormente se selecciona el botón Create dentro de la ventana que aparece, se asigna un nombre a la biblioteca nueva y se confirma la acción (Ver Figura 7.3):

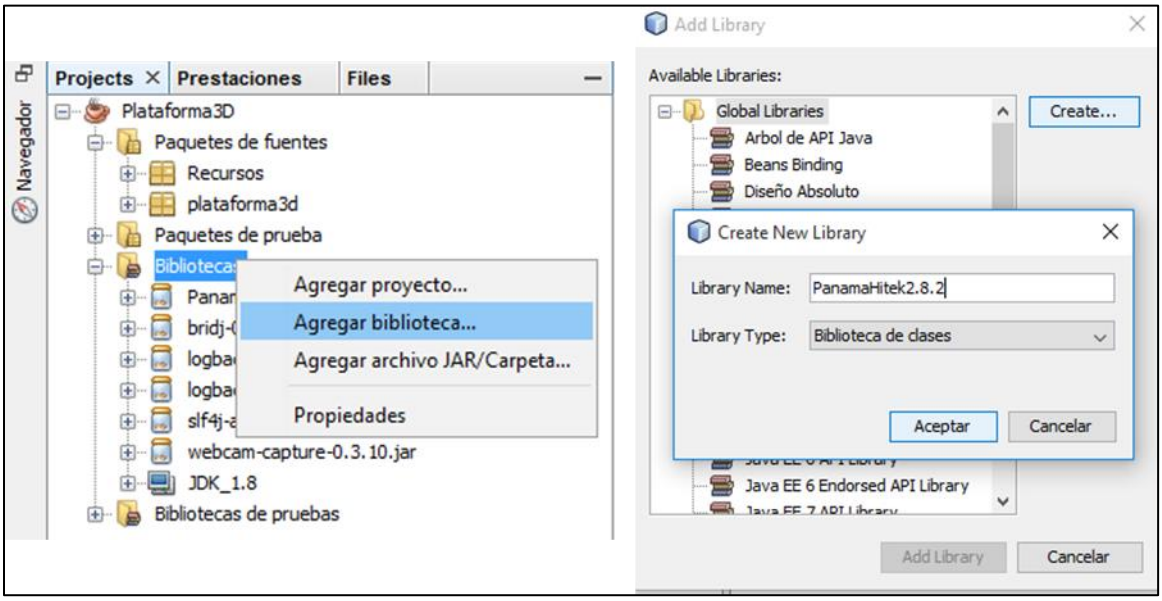

Figura 7.3 Agregar biblioteca nueva.

Fuente: Elaboración propia.

Finalmente se da clic en el botón Agregar archivo JAR/Carpeta en la ventana que aparece y se selecciona el archivo con extensión .jar descargado desde la página de PanamaHitek que contiene la librería y se agrega ésta al proyecto (Ver Figura 7.4):

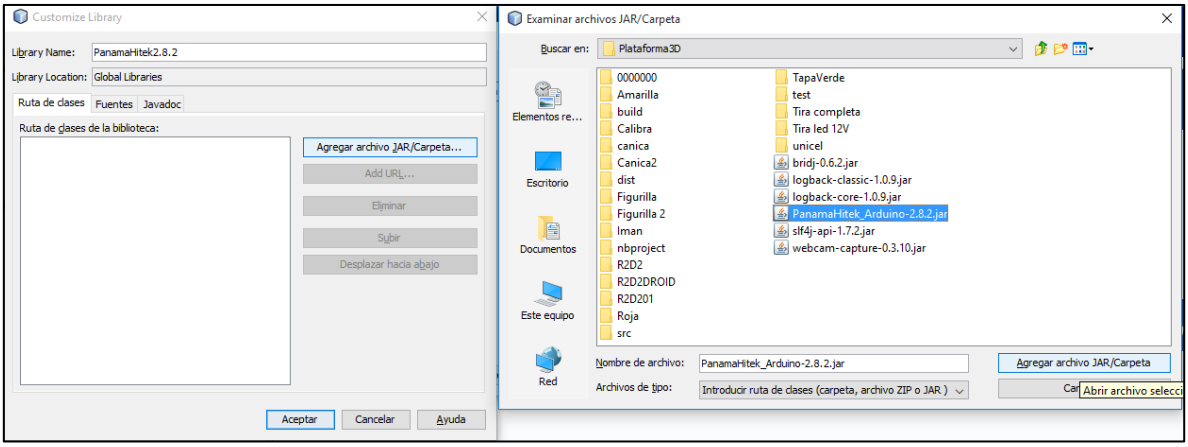

Figura 7.4 Agregar el archivo .jar a la biblioteca nueva.

Fuente: Elaboración propia.

La conexión con Arduino se realiza creando un objeto de la clase Arduino dentro de la clase principal como se muestra en el Ejemplo 5 y proporcionándole los parámetros, como lo son el identificador del puerto y el valor de baudios, de la placa con la que se está trabajando mediante el método arduinoTX(), este método se utiliza para iniciar la conexión de Java con Arduino en modalidad de transmisión de datos (Ejemplo 6).

```
PanamaHitek_Arduino Arduino = new PanamaHitek_Arduino();
```

Ejemplo 5. Creación de un objeto de la clase Arduino.

Fuente: Elaboración propia.

```
Arduino.arduinoTX("COM4",9600);
```

Ejemplo 6. Paso de parámetros correspondientes a la placa en uso (Puerto,Baudios).

Fuente: Elaboración propia.

El puerto puede ser comprobado dentro del IDE de Arduino mediante la opción Puerto del menú de Herramientas (ver Figura 7.5) mientras que el valor de baudios más comúnmente utilizado es de 9600.

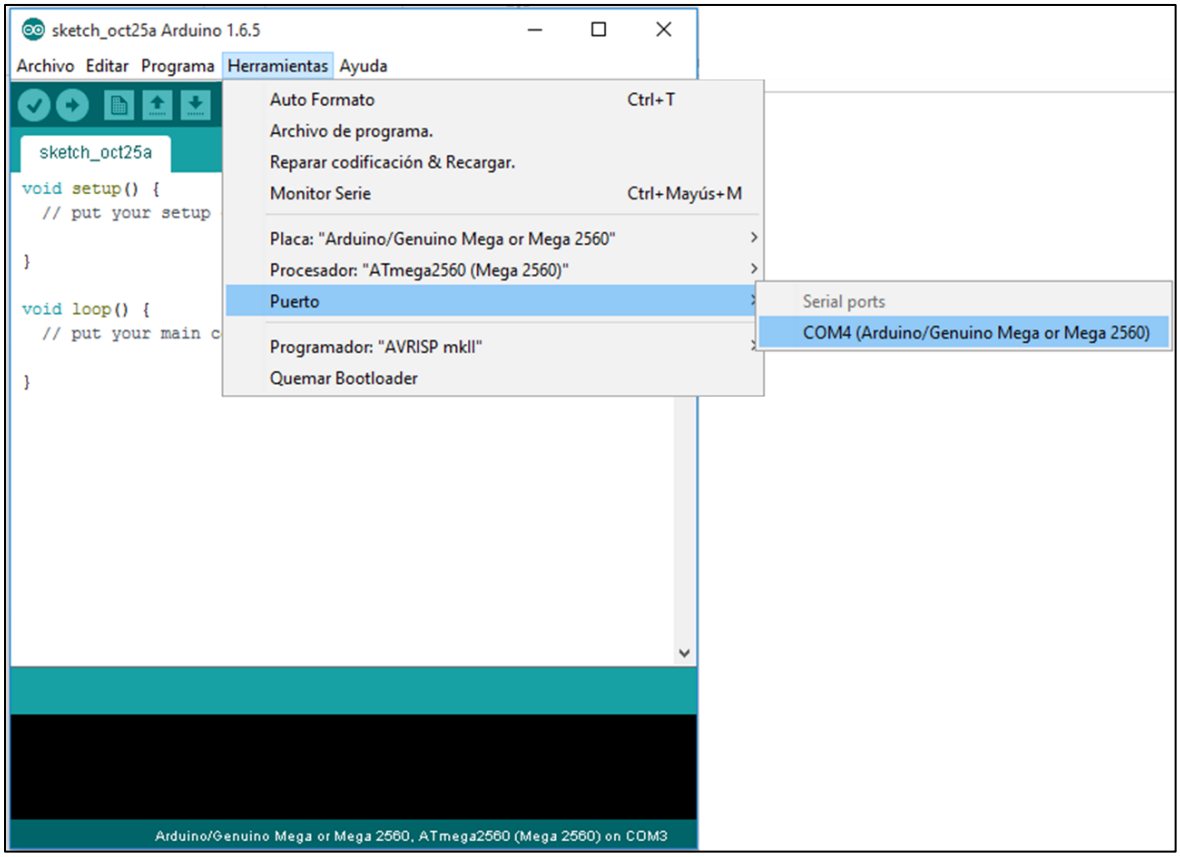

Figura 7.5 Identificación del puerto en uso dentro del IDE de Arduino.

Fuente: Elaboración propia.

Control de movimiento

El proceso de control de movimiento se lleva a cabo en dos etapas, durante la primera de ellas se envía un dato a la placa Arduino mediante el puerto USB que le da la instrucción de girar la plataforma una posición. Dado que este proceso debe ser ejecutado repetidas veces es lanzado dentro de un ciclo for acompañado de la instrucción mediante la cual se captura la fotografía del objeto en esa posición seguida de una pausa de 1 segundo para que la computadora procese y

guarde dicha imagen. Al finalizar un total de 20 repeticiones se envía un nuevo dato a la placa Arduino que indica que la cámara esta lista para ser elevada una posición y comenzar de nuevo el ciclo de captura. Este proceso se ejecuta dos veces para conseguir que la cámara alcance su posición más alta antes de iniciar el último ciclo de captura. El código encargado de esta tarea se ejemplifica a continuación (Ver Ejemplo 7):

```
    for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 20; j++) {
               x++;
               Foto(x);
               try {
               Thread.sleep (1000);
               } catch (Exception e) {
               // Mensaje en caso de que fallas
               }
            enviaDato("1"); //Dato enviado a Arduino para identificar que se solicita la rotación
de la plataforma.
            }
            enviaDato("2"); //Dato enviado a Arduino para solicitar que la cámara se eleve una
posición.
          }
```

Ejemplo 7. Primera etapa del código encargado de controlar el movimiento de los motores.

Fuente: Elaboración propia.

Finalmente, durante la última etapa de movimiento se lleva a cabo un último ciclo de captura y se envía mediante el puerto USB un último carácter que indica a Arduino que puede devolver la cámara a su estado inicial. Dicho ciclo se ejemplifica a continuación (Ver Ejemplo 8):

```
    for (int j = 0; j < 20; j++) {
            enviaDato("1"); //Dato enviado a Arduino para identificar que se solicita la rotación
de la plataforma.
            x++;
            Foto(x);
            try {
               Thread.sleep (1000);
               } catch (Exception e) {
               // Mensaje en caso de que falle
               }
          }
          enviaDato("3"); //Dato enviado a Arduino para solicitar que la cámara regrese a su
posición original.
```

Ejemplo 8. Segunda etapa del código encargada de controlar el movimiento de los motores.

Fuente: Elaboración propia.

Interfaz de visualización y captura de objeto

Para esta tarea se utilizó la librería webcam-capture-0.3.10, la cual permite el uso de cámaras web incorporadas o externas conectadas mediante USB. Esta librería puede ser adquirida desde su página oficial (Firyn, 2017) misma en la que se pueden revisar un gran número de ejemplos y recomendaciones para su uso.

Nuevamente el uso de esta se hace mediante dos etapas, la primera de ellas se encarga de crear una pantalla que funciona como visor en tiempo real del objeto al que está enfocando la cámara, además de permitir la posibilidad de definir previamente la fuente de imagen (cámara) deseada.

Inicialmente se creó una clase de nombre visor y esta fue inicializada desde la clase principal de java para poder llamarla a ejecución desde ésta.

Ya en la clase visor se declaran tres objetos de suma importancia para el uso de las características de la interfaz, la webcam, un panel sobre el que se hará la visualización y un webcampicker que funcionará como selector de la fuente de video.

El picker es llenado mediante un método addItemListener() que se encarga de buscar e identificar todos los dispositivos de imagen conectados a la computadora, posteriormente el objeto webcam

toma la referencia de la primer cámara que identifica y la imagen que esta capta es cargada directamente sobre el panel a manera de video en tiempo real (Figura 7.6).

Figura 7.6 Video en tiempo real de un objeto a partir de la clase visor.

Fuente: Elaboración propia.

Además, el WebCamPicker al obedecer a su método listener() detecta el cambio en la selección de cámara por lo que, si se desea, se puede cambiar el video mostrado en el panel por el de cualquier otra cámara identificada en cualquier momento del proceso si se contara con más de una y la actual no fuera la correcta para llevar a cabo la tarea de captura (Figura 7.7).

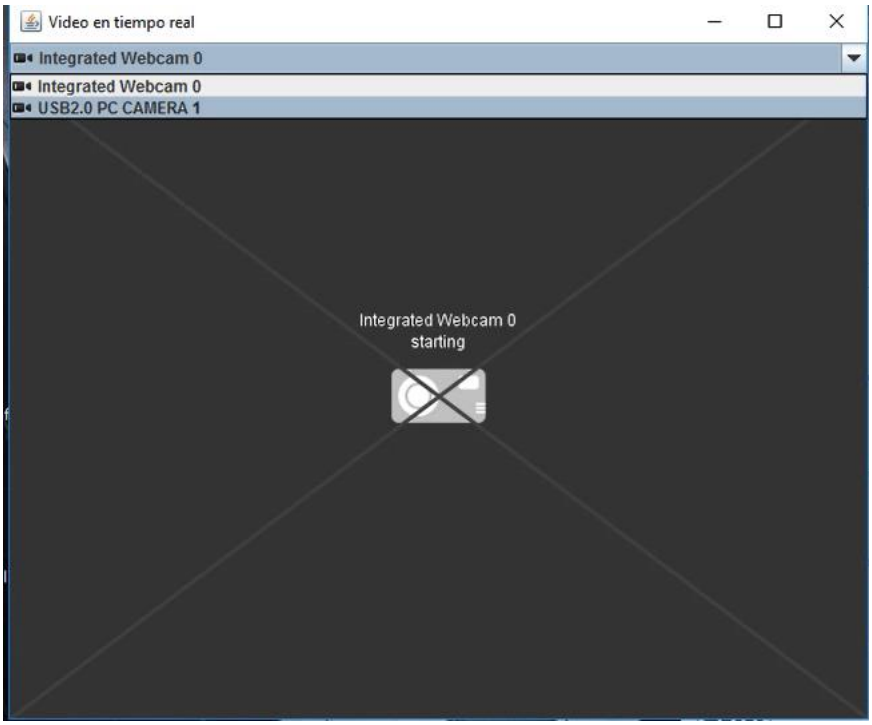

Figura 7.7 Detección de diferentes dispositivos de imagen mediante el WebCamPicker.

Fuente: Elaboración propia.

Implementación del Script MeshRecon

MeshRecon funciona como una herramienta de reconstrucción de mallas para modelado 3D basada en el principio de SFM (Structure From Motion), es decir, la estimación de la estructura en tres dimensiones de un objeto a partir de un conjunto de imágenes 2-D de este (Kang, 2015). Esta técnica es utilizada en varios campos, desde la reconstrucción 3D hasta la producción de aplicaciones para realidad aumentada.

SFM puede ser tratado de diferentes maneras, dependiendo del objetivo del problema es que se deben ajustar diferentes factores como el número y tipo de cámaras o el orden en que las imágenes son tomadas. Si las imágenes son tomadas a partir de una sola cámara calibrada, la estructura 3D generada es recuperada a escala, es decir, se puede reescalar la estructura y movimiento de la cámara y aun así mantener las observaciones del modelado (MathWorks, 2017).

Para el caso de MeshRecon, éste se apoya en el sistema de reconstrucción de VisualSFM, mismo que explota las características de paralelismo multinucleo incorporado mediante la tecnología CUDA (Compute Unified Device Architecture/Arquitectura Unificada de Dispositivos de Cómputo), la cual es una arquitectura propia de las tarjetas NVidia que aprovecha la gran potencia de la GPU (Graphic Process Unit/Unidad de Procesamiento Grafico) para potenciar un incremento al rendimiento del sistema en aplicaciones gráficas (NVidia, 2017).

VisualSFM trabaja el proceso de reconstrucción en cuatro etapas, durante la primera se carga al software el conjunto de imágenes del objeto a procesar a partir de las cuales se generara el modelo 3D (Figura 7.8) :

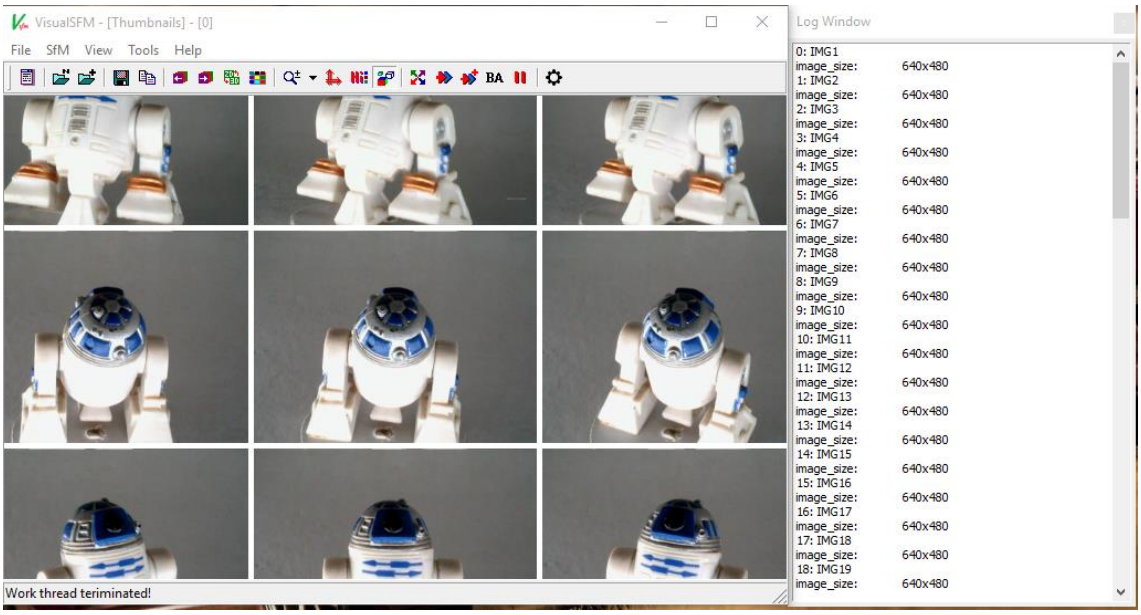

Figura 7.8 Etapa de adquisición de imágenes desde el software VisualSFM.

Fuente: Elaboración propia.

Durante la segunda etapa se lleva a cabo el emparejamiento de imágenes con la finalidad de que el software detecte la correspondencia entre ellas (Figura 7.9):

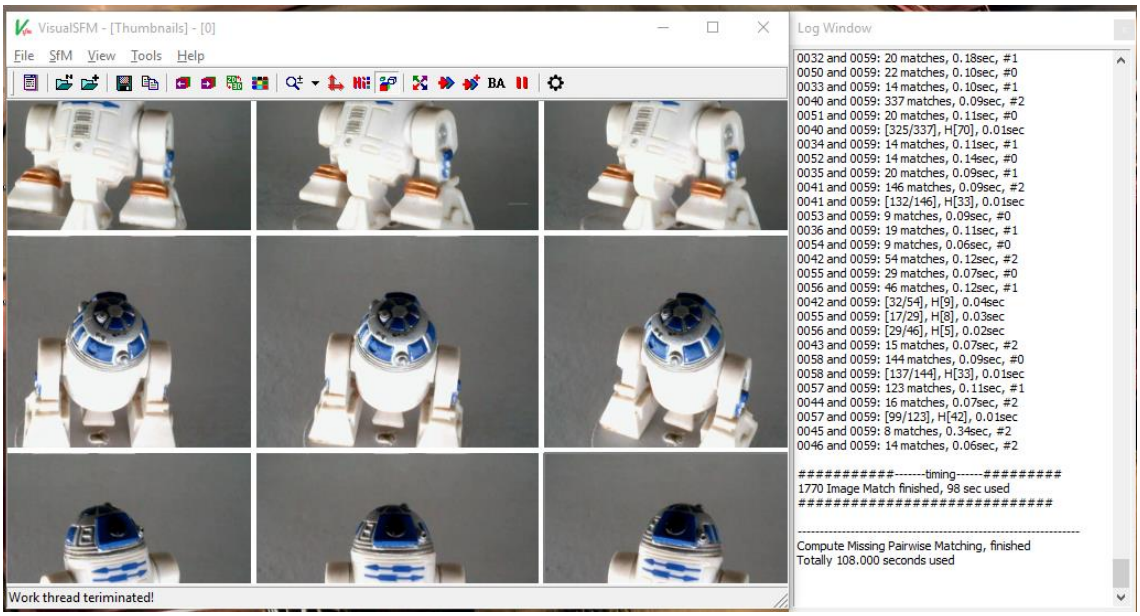

Figura 7.9 Etapa de emparejamiento de imágenes mediante el software VisualSFM.

Fuente: Elaboración propia.

Para la tercer etapa, ya con las imágenes emparejadas y sus correspondencias identificadas el software se encarga de generar la nube de puntos correspondientes al objeto, es decir, triangula dentro de un entorno en tres dimensiones los puntos que detecta como correspondientes entre imágenes a fin de generar una vista previa de la malla que compone al objeto como se muestra en la Figura 7.10:

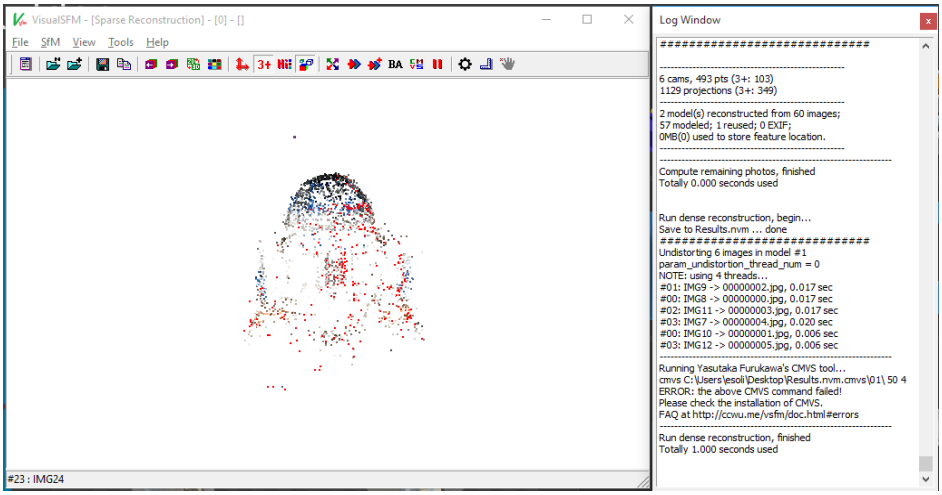

Figura 7.10 Etapa de identificación de la nube de puntos.

Fuente: Elaboración propia.

Finalmente MeshRecon se encarga de generar a partir de esta nube de puntos el modelo 3D del objeto con que se está trabajando, este modelo puede ser visualizado desde aplicaciones de diseño 3D tales como 3D Builder (Figura 7.11):

Figura 7.11 Modelo generado por MeshRecon a partir de la nube de puntos proporcionada por VisualSFM.

Fuente: Elaboración propia.

Gracias al script proporcionado dentro del paquete de MeshRecon este proceso puede ser automatizado únicamente ajustando algunos parámetros dentro de este. Dicho script puede ser adquirido desde la página del proyecto pero su codificación se muestra en el Ejemplo 9:

```
    cd vsfm
    .\VisualSFM.exe sfm ..\img ..\results\output.nvm
    cd..

    cd meshrecon
    .\nvm2sfm.exe ..\results\output.nvm ..\results\output.sfm 1
    .\MeshRecon_init.exe ..\results\output.sfm ..\results\MESH_init.ply
    .\MeshRecon_refine.exe ..\results\output.sfm ..\results\MESH_init.ply
..\results\MESH_refine.ply 0
    cd ..
```

Ejemplo 9. Contenido del Script de reconstrucción 3D de MeshRecon.

Fuente: Elaboración propia.

Este conjunto de instrucciones son ejecutadas en un archivo .bat (Archivo de procesamiento por lotes) y básicamente se encarga de ejecutar el software VisualSFM desde su ruta de origen y pasarle como parámetros la carpeta que contiene el conjunto de imágenes obtenidas a partir del prototipo y la carpeta de salida donde será almacenado el modelo final. Una vez obtenida la nube de puntos desde VisualSFM, MeshRecon se encarga de tomarla y generar a partir de esta el modelo final mediante la técnica SFM y guardar este resultado en un archivo con nombre MESH_refine.ply.

Interfaz Principal

La interfaz de usuario fue diseñada de manera que fuera lo más intuitiva y llamativa para el usuario, en esta se muestran únicamente 5 botones principales y un campo donde ingresar el nombre con el que se desea etiquetar al modelo que se está generando (ver Figura 7.12).

Figura 7.12 Interfaz principal del módulo de software.

Fuente: Elaboración propia.

La función de cada uno de los botones es la siguiente:

Calibrar enfoque manualmente: Está encargado de abrir la clase visor, es decir, la interfaz de visualización en tiempo real de la cámara. Esto con el fin de permitir al usuario colocar el objeto en la posición y con el enfoque correcto antes de comenzar el proceso de captura.

Realizar captura de prueba: Permite realizar una captura de prueba, a fin de verificar que la cámara esté funcionando de manera adecuada.

Probar motores: Permite comprobar el funcionamiento de los motores. Al presionar el botón, los motores ejecutan todos los ajustes de movimientos necesarios para llevar a cabo un ciclo de captura pero sin obtener ninguna imagen.

Iniciar sesión fotográfica: Comienza el proceso de captura.

Comenzar reconstrucción 3D: Una vez adquirido el conjunto de imágenes del objeto deseado, éste botón permite llevar a cabo el proceso de modelado de manera autónoma mediante las herramientas incluidas de manera nativa dentro del software (MeshRecon).

Capítulo 8 Pruebas y resultados

El prototipo fue sometido a una serie de pruebas con la finalidad de medir el tiempo necesario para llevar a cabo la captura de un total de 60 imágenes, que es el recomendado por la mayoría de alternativas de modelado, y el tiempo de procesamiento para generar un modelo 3D a partir de dicho conjunto de imágenes. Los resultados iniciales indicaron que el prototipo tarda un aproximado de dos minutos en tomar el conjunto de 60 fotografías.

Cada una de las imágenes capturadas tiene un peso de almacenamiento de entre 15 y 35 KB dando como resultado un promedio aproximado de 1.50 MB de almacenamiento para el conjunto de imágenes final. Para cada prueba el conjunto de imágenes resultante fue sometida a procesamiento por medio de diferentes alternativas de software de modelado y dichos resultados fueron comparados entre sí a fin de identificar la ventaja y desventaja de cada uno respecto al software incorporado de manera nativa dentro de este proyecto (MeshRecon).

A continuación se muestran los resultados obtenidos a partir de algunas de las pruebas realizadas sobre dos diferentes tipos de flor y una figurilla de plástico con diferentes características y tamaños.

Flor amarilla

La primer serie de pruebas fue llevada a cabo sobre una flor silvestre de aproximadamente 2 cm de diámetro por 2.5 cm de largo, esta fue colocada sobre la plataforma giratoria mediante un soporte improvisado con la finalidad de mantenerla estática y en una posición que facilitara la captura del mayor número de características de esta como lo son la forma de las hojas interiores y exteriores y la figura del tallo (ver Figura 8.1).

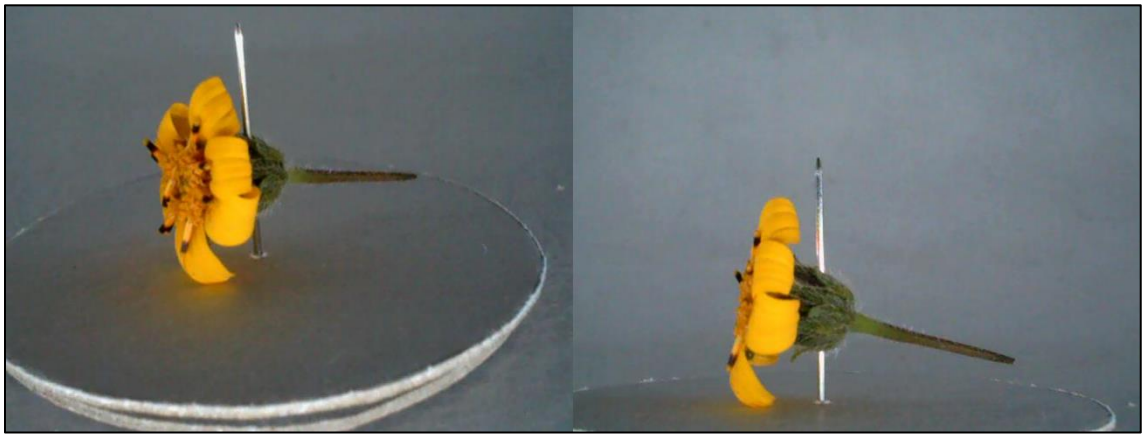

Figura 8.1 Flor amarilla usada durante la primera prueba del prototipo.

Fuente: Elaboración propia.

El tamaño necesario para almacenar las capturas de este objeto oscila entre los 16 y los 26 KB. Por lo que el tamaño total del conjunto de imágenes obtenido es de 1.39 MB. Una vez comprobado el conjunto de imágenes correspondientes a la primera prueba, stas fueron procesadas mediante el uso de tres diferentes alternativas de modelado.

Para el caso del software PhotoScan, cuyo procesamiento se ejecuta de manera local, el tiempo de procesamiento aproximado fue de 10 minutos con un tamaño de archivo final de 12.2 MB usando una configuración de procesado de alta precisión. En la Figura 8.2 se muestra el resultado obtenido a partir de esta propuesta:

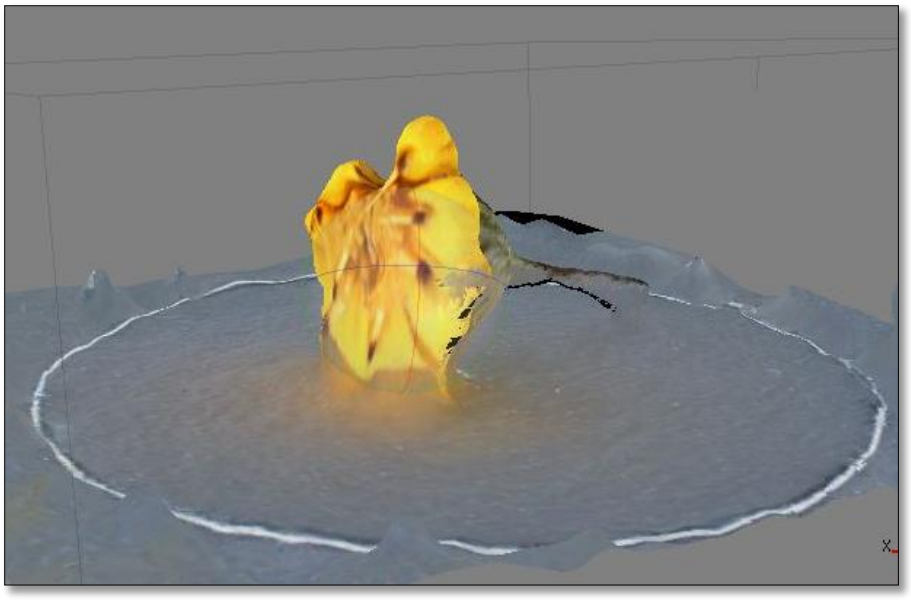

Figura 8.2 Resultado obtenido durante la primera prueba mediante el uso de PhotoScan.

Fuente: Elaboración propia.

En el resultado a partir de PhotoScan se puede apreciar un modelado de calidad regular a pesar de haber usado la configuración de alta precisión. Además, es evidente que a pesar de llevarse a cabo la captura en un entorno controlado, aún se observa ruido o distorsión durante el modelado del objeto.

Posteriormente, el mismo conjunto de imágenes fue procesado mediante el software ReMake cuyo procesamiento es llevado a cabo de manera remota. El tiempo aproximado de procesado del conjunto de imágenes fue de 15 minutos con un tamaño de archivo final de 1.47 MB. En la Figura 8.3 se muestra el resultado obtenido a partir de dicha propuesta:

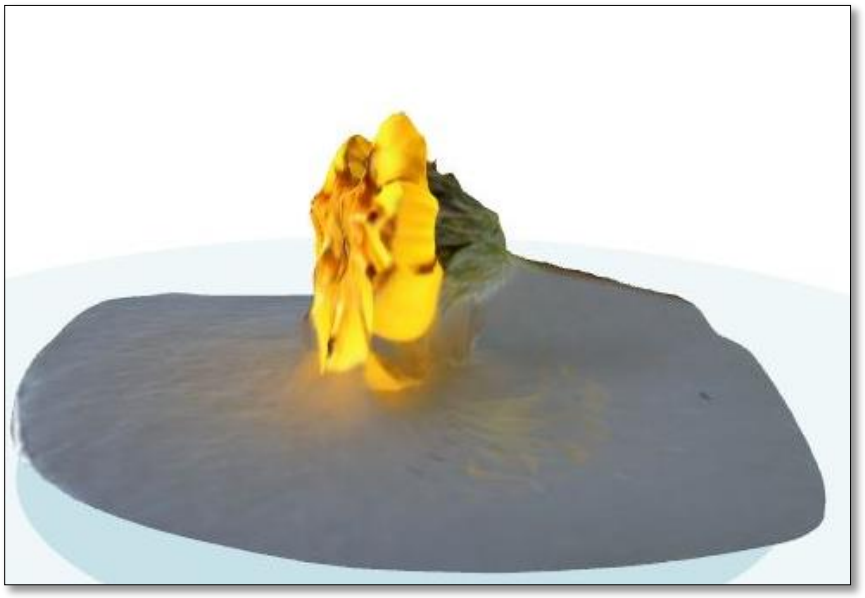

Figura 8.3 Resultado obtenido durante la primera prueba mediante el uso de ReMake.

Fuente: Elaboración propia.

Como se puede observar, la precisión mejora mucho con respecto al resultado obtenido a partir de PhotoScan. Se observa mucha menor distorsión en el modelado, excepto por el tallo, y una correcta simulación de la textura del objeto. Sin embargo, el tiempo de procesamiento es considerablemente mayor.

Finalmente, el conjunto de imágenes fue procesado mediante el software que se integra de manera nativa dentro de la paquetería de este proyecto, el tiempo de procesamiento aproximado fue de 9 minutos con un tamaño de archivo final de 616 KB. El resultado obtenido a partir de este procesamiento se muestra a continuación en la Figura 8.4:

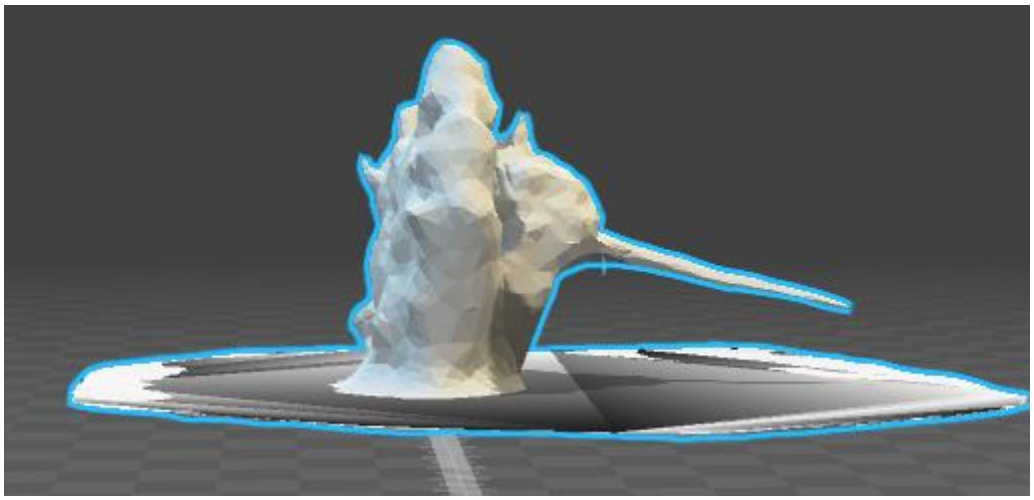

Figura 8.4 Resultado obtenido mediante el uso de MeshRecon.

Fuente: Elaboración propia.

Como se puede observar, a pesar de carecer de textura, el modelado resulta ser bastante preciso y con poca distorsión respecto a los bordes del objeto.

R2D2

La segunda serie de pruebas se llevó a cabo utilizando una figurilla de plástico del personaje de ciencia ficción conocido como "R2D2". Este objeto tiene un tamaño aproximado de 5*3*3 cm y una textura semi reflejante (ver Figura 8.5).

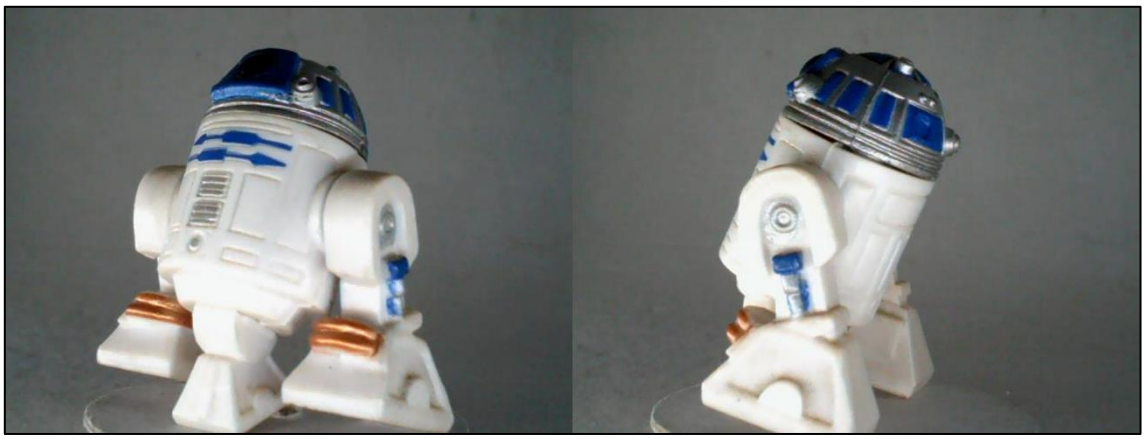

Figura 8.5 Figurilla de plastico R2D2 usada durante la segunda serie de pruebas del prototipo.

Fuente: Elaboración propia.

El tamaño necesario para realizar cada captura de este objeto oscila entre 19 y 26 KB, por lo que el tamaño total del conjunto de imágenes fue de 1.41 MB.

Mediante PhotoScan, el tiempo aproximado de procesamiento fue de 14 minutos con un tamaño de archivo final de 13.3 MB. El resultado obtenido se muestra en la Figura 8.6:

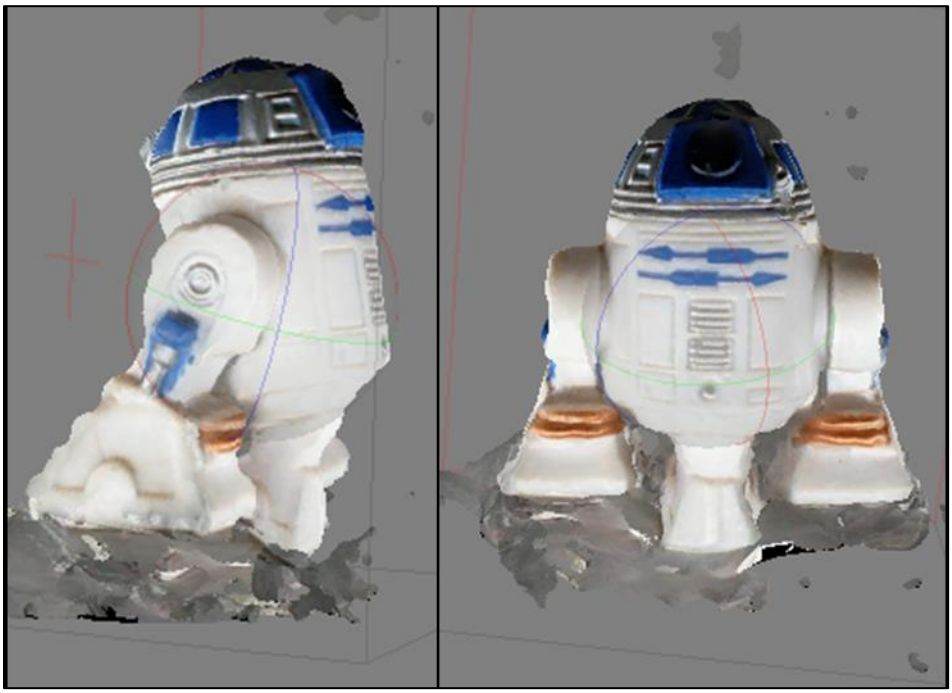

Figura 8.6 Resultado obtenido durante la segunda prueba mediante el uso de PhotoScan.

Fuente: Elaboración propia.

Nuevamente se puede apreciar un alto grado de distorsión en el modelado a pesar de que la configuración usada fue la de alta precisión, sin embargo, la simulación de textura del objeto resulto ser bastante buena.

Posteriormente, el objeto fue procesado usando el software ReMake, cuyo tiempo aproximado de procesamiento fue de 17 minutos con un tamaño final de archivo de 2.1 MB. El resultado se muestra en la Figura 8.7:

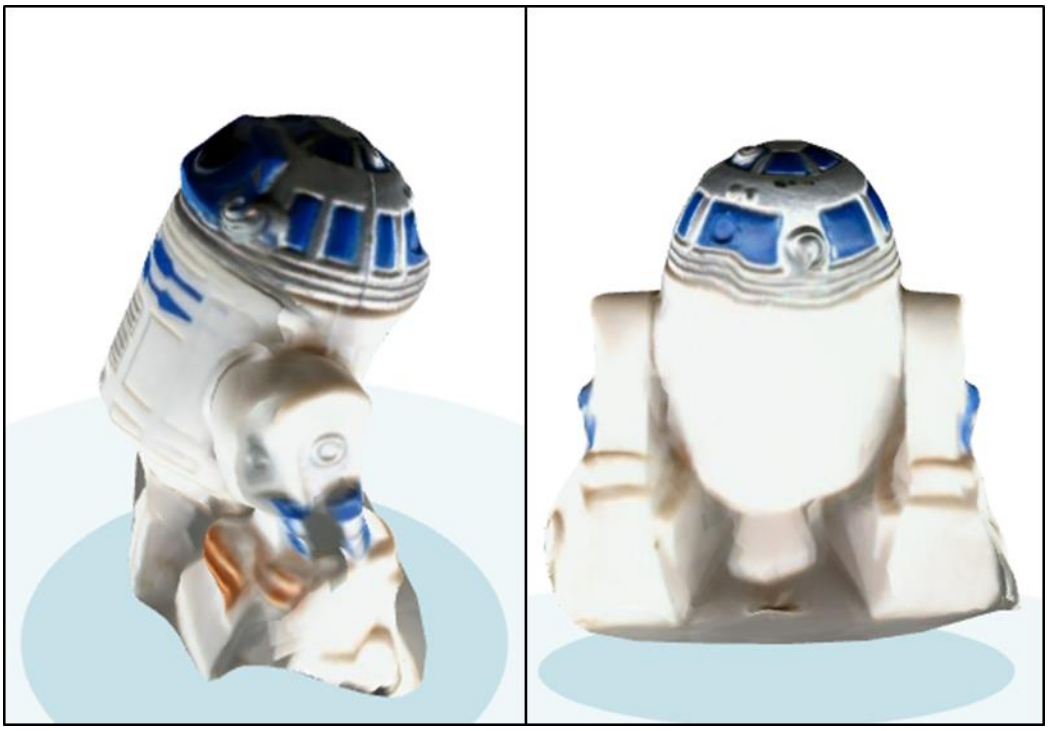

Figura 8.7 Resultado obtenido durante la segunda prueba mediante el uso de ReMake.

Fuente: Elaboración propia.

A diferencia de la prueba anterior en esta ocasión ReMake arrojo mucha más distorsión en las partes inferiores del modelado generando así una deficiencia también en la adquisición y simulación de la textura del objeto.

Finalmente, el conjunto de imágenes fue procesado mediante MeshRecon arrojando un tiempo de procesamiento de 9 minutos y un archivo de tamaño final de 588 KB. El modelo resultante se muestra en la Figura 8.8:

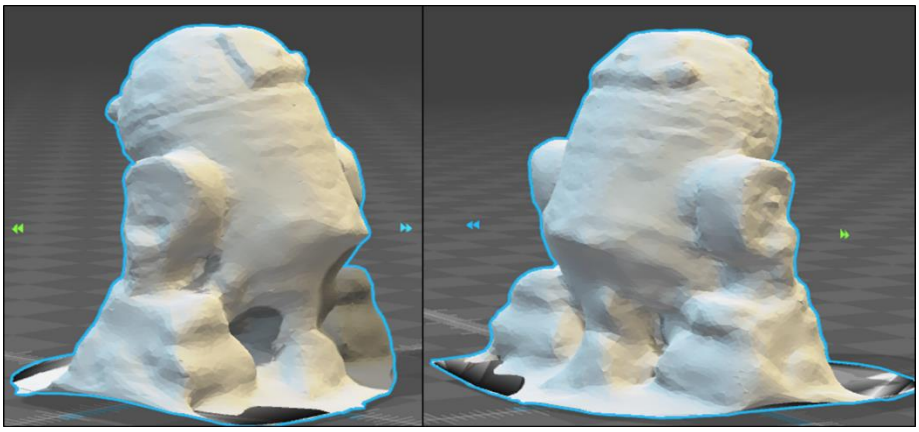

Figura 8.8 Resultado obtenido durante la segunda prueba mediante el uso de MeshRecon.

Fuente: Elaboración propia.

Como se puede observar en la figura anterior la calidad de modelado es muy buena y casi no se detectaron señales de distorsión en él.

Flor Roja

La tercer serie de pruebas llevadas a cabo se realizaron modelando otro tipo de flor con un tamaño aproximado de 3 cm de diámetro por 3 cm de largo y una textura opaca (ver Figura 8.9).

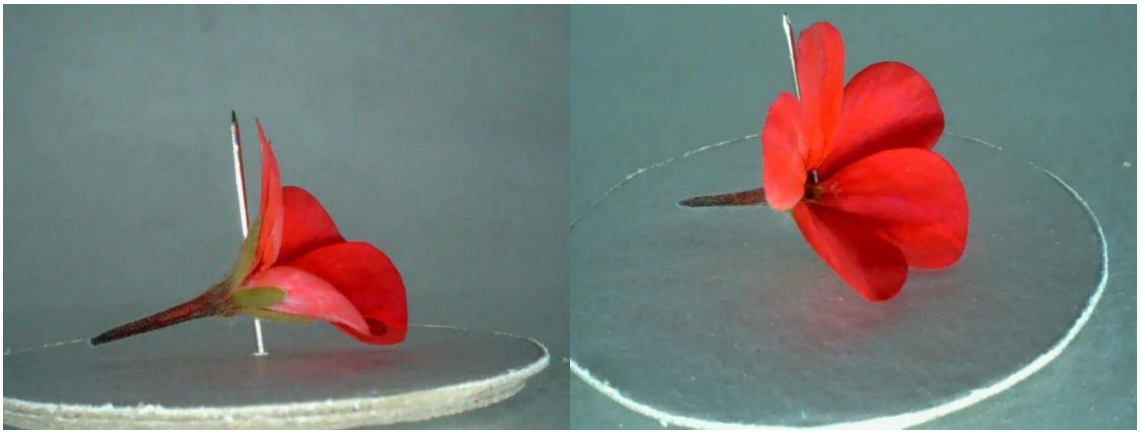

Figura 8.9 Flor roja usada durante la tercera serie de pruebas del prototipo.

Fuente: Elaboración propia.

El tamaño necesario para capturar cada imagen de esta flor oscila entre 20 y 30 KB y un tamaño total del conjunto de imágenes fue de 1.52 MB.

Mediante PhotoScan el tiempo aproximado de procesamiento fue de 12 minutos con un tamaño de archivo final de 12.2 MB. El resultado obtenido se muestra a continuación en la Figura 8.10:

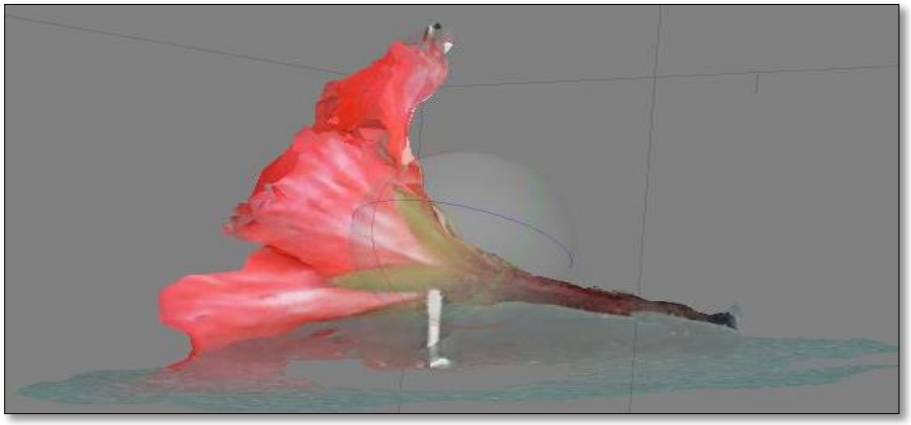

Figura 8.10 Resultado obtenido durante la tercera prueba mediante el uso de PhotoScan.

Fuente: Elaboración propia.

A comparación de los dos ejemplos anteriores llevados a cabo con PhotoScan, en éste modelado se puede ver menos distorsion del objeto y una correcta simulación de la textura del objeto.

Para el caso de ReMake, el tiempo aproximado de procesamiento fue de 19 minutos y un tamaño final de archivo de 1.34 MB. El resultado se muestra en la Figura 8.11:

Figura 8.11 Resultado obtenido durante la tercera prueba mediante el uso de ReMake.

Fuente: Elaboración propia.

El resultado obtenido a partir de ReMake demostró ser de muy buena calidad y precisión, no se detecta distorsión importante en este y la simulación de textura del objeto es excelente.

Finalmente, el conjunto de imágenes fue procesado mediante MeshRecon, dando como resultado un tiempo de procesamiento de 8 minutos para un tamaño de archivo final de 456 KB. El resultado se muestra en la Figura 8.12:

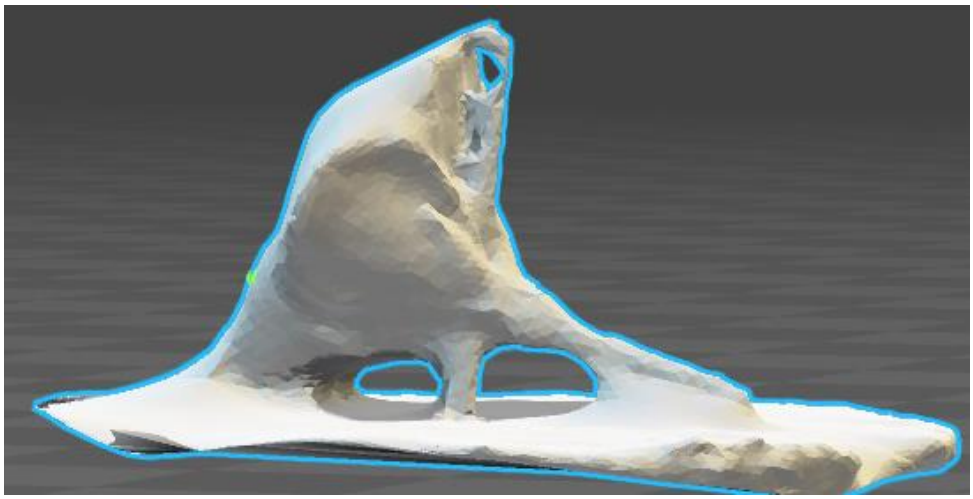

Figura 8.12 Resultado obtenido durante la tercera prueba mediante el uso de MeshRecon.

Fuente: Elaboración propia.

Lamentablemente para la última prueba MeshRecon no resulto ser tan eficiente como en los ejemplos anteriores, se detecta un exceso de distorsión en el modelado lo cual dificulta la identificación a primera vista del objeto.

Capítulo 9 Conclusiones

El entorno de captura

El prototipo final tiene la facilidad de ser armado y desamado con sencillez y rapidez cumpliendo así con el objetivo facilitar el transporte de la estructura para evitar la adquisición de un ejemplar del prototipo para cada espacio de trabajo donde se requiera llevar a cabo la digitalización. En ensamblaje del prototipo no toma más de 5 minutos tomando en cuenta desde la colocación de cada pieza en su posición hasta la conexión de los componentes del mismo.

Cabe mencionar que para el ejemplar actual del prototipo los objetos a modelar pueden tener un tamaño máximo de 5 cm de alto y 5 cm de ancho. Sin embargo, este tamaño puede ser ampliado si el soporte de la cámara se aleja más del centro de rotación para ampliar la zona de captura o se sustituye la cámara con una de mayor apertura de foco.

Finalmente el fondo curvo o fondo infinito soluciona el problema de líneas y marcas encontradas en la captura ocasionadas por las uniones entre las paredes del dispositivo reduciendo así la dificultad de identificación del objeto y aumentando la calidad del modelado.

Hardware

Las plataformas libres permiten una reducción de costos y la adaptación de los componentes a las necesidades particulares del proyecto. Alternativas como Arduino permite implementar el uso de la electrónica y la programación de sistemas embebidos para desarrollar proyectos con fines multidisciplinarios en una amplia gama de campos de aplicación.

Además, pueden ser usados en entornos de aprendizaje incluso con niños desde edades tempranas permitiendo así capacitarlos y orientarlos en la disciplina de la electrónica, mismos

que pueden ir evolucionando sus diseños con los diversos modelos que estas alternativas proveen estimulando sus capacidades y creatividad a fin de conseguir diseñar sus propias tecnologías a precios muy accesibles.

Pruebas con alternativas de software comercial contra el software nativo

A continuación se presenta una tabla en la cual se resumen de manera ordenada los resultados obtenidos a partir de cada una de las series de pruebas:

Software	**Ejemplo**	**Tiempo de captura**	**Tamaño promedio de Imagen**	**Tipo de Procesamiento**	**Tiempo de procesamiento**	**Tipo de Archivo Final**	**Tamaño de Archivo final**
PhotoScan	Flor Amarilla	2 Minutos	21.6 KB 640x480 px	Local	10 Minutos	.psx	12.2 MB
	Flor Roja		25.8 KB 640x480 px		12 Minutos		12.2 MB
	Figurilla de Plástico		22.1 KB 640x480 px		14 Minutos		13.3 MB
ReMake	Flor Amarilla		21.6 KB 640x480 px	Remoto	15 Minutos	.rcm	1.47 MB
	Flor Roja		25.8 KB 640x480 px		19 Minutos		1.34 MB
	Figurilla de Plástico		22.1 KB 640x480 px		17 Minutos		2.1 MB
MeshRecon	Flor Amarilla	2 Minutos	21.6 KB 640x480 px	Local	9 Minutos	.ply	616 KB

(Software Nativo)	Flor Roja		25.8 KB 640x480 px		8 Minutos		456 KB
	Figurilla de Plástico		22.1 KB 640x480 px		9 Minutos		588 KB

Para el caso de ReMake, y tomando en cuenta que la velocidad de subida con la que se realizaron las pruebas, al momento de cargar las imágenes al servidor de Autodesk fue de 0.25 Mbps, y dado que el tamaño total de la galería es de 1.52 MB, el tiempo de subida estimado es de (1.52/.25) = 6.08 minutos. Sin embargo, cabe mencionar que la velocidad de la red puede variar dependiendo del uso de ésta y la saturación de información de la misma, por lo cual este tiempo puede verse incrementado.

En retrospectiva el tiempo de procesamiento de MeshRecon fue el menor respecto a los de sus contrincantes, mientras que la calidad del modelado demostró ser mejor para el caso de ReMake. Sin embargo, en la mayoría de estas MeshRecon demostró también ser bastante eficiente, más incluso que PhotoScan.

Software

Esta propuesta representa un medio viable para automatizar del proceso de digitalización de objetos 3D de una manera sencilla e intuitiva para cualquier tipo de usuario y en un entorno de software que puede ser ejecutado en diferentes equipos de cómputo. El uso de Java permite el desarrollo y distribución de aplicaciones en los campos de la programación móvil e incrustada, videojuegos, contenido web y software empresarial con la posibilidad de generar interfaces que resulten atractivas al usuario y con diseños estilizados según su finalidad.

Los tiempos promedio de respuesta para generar una digitalización completa son de 15 minutos con 60 imágenes y un peso de archivo final de entre 1 y 12 Mb dependiendo del software.

El resultado final puede ser editado o manipulado por diferentes tipos de Software de diseño 3D permitiendo así mejorar este aplicándole algún tipo de corrección o capa de color si así se desea.

Conclusiones generales y costos de producción

En la tabla siguiente se resumen los costos de producción del prototipo tal y como se presenta en este documento:

Elemento	**Costo Unitario**
Estructura de Triplay	$150.00
Placa Arduino Mega 2560	$740.00
Motor a pasos 28BYJ-48 (x2)	$150.00
Cámara web Ele-gate HD 18 Mpx	$180.00
PlastiAcero	$35.00
Pintura Acrilica Blanco Mate (x2)	$45.00
Driver Motor a Pasos (x2)	$124.00
Tira de Led	$30.00
TOTAL	$1014.00 MN

La reducción de costos actual frente a un scanner 3D comercial es de un aproximado de $13,500 MN comparado con el escáner de mano Sense de la compañía 3D Systems con lo

cual se demuestra que el objetivo de producir un prototipo funcional y a bajo costo se cumplió por completo. Debe tomarse en cuenta que el costo final puede verse reducido aún más reemplazando el modelo actual de Arduino por uno más limitado siempre y cuando este no comprometa los requerimientos para el funcionamiento del dispositivo.

Anexos

Comprobante de envió de artículo científico para su publicación en la Revista Iberoamericana de las Ciencias Computacionales e Informática.

Revista
Iberoamericana de las
Ciencias Computacionales
e Informática
ISSN: 2007 - 9915

Enrique Solis Cielo.

Presente

La Revista Iberoamericana de las Ciencias Computacionales e Informática RECI, a través de la presente le informa que el trabajo titulado **"PROTOTIPO PARA AUTOMATIZAR LA DIGITALIZACIÓN DE OBJETOS EN TRES DIMENSIONES"** De la autoría de **Enrique Solis Cielo, Oziel Lugo Espinosa, Joel Ayala de la Vega y Alfonso Zarco Hidalgo** turnado al comité editorial para evidenciar la originalidad del escrito ha dado como resultado ***LA ORIGINALIDAD DEL ESCRITO*** y por lo tanto puede continuar con el proceso de revisión por pares.

A partir del día 26 de Septiembre del 2017 se inicia el proceso de evaluación mismo que, no deberá de exceder los 60 días para que el comité editorial reciba las posibles observaciones al artículo y se haga de su conocimiento mediante carta membretada.

Se extiende la presente a petición del interesado, para los efectos legales y formales que convengan.

ATENTAMENTE

Guadalajara, Jalisco a 26 de Septiembre del 2017

Dr. Francisco Santillán Campos

Director Editor

Av. México #2798 piso 5-B Torre Diamante Col. Circunvalación Vallarta. C.P. 44680 www.reci.org.mx
Tel: 044 3310618187 Guadalajara, Jalisco, México

Referencias

A. Perez, O., A. Ruiz de Olano, A., & J. Perez, A. (2006). Una metodología para el desarrollo de hardware y software embebidos en sistemas críticos de seguridad. *Sistemas, cibernética e informática.*, 70-75.

Agisoft. (2017). *Agisoft*. Obtenido de Agisoft: http://www.agisoft.com/

Arduino. (2017). Obtenido de https://www.arduino.cc/

Attaway, S. (2016). *Matlab: A Practical Introduction to Programming and Problem Solving.* Boston: Elsevier.

Autodesk. (2015). *Autodesk ReMake*. Obtenido de http://remake.autodesk.com/try-remake#system-requirements

Banzi, M. (2011). *Getting Started with Arduino.* O'Reilly Media, Inc. .

Caicedo Pedrera, A. (2017). *Arduino para principiantes: Segunda Edición.* IT Campus Academy.

Cline, L. S. (2014). *3D Printing with Autodesk 123D, Tindercad and MakerBot.* McGraw Hill Professional.

Corona Ramírez, L. G., Abarca Jiménez, G. S., & Mares Carreño, J. (2014). *Sensores y actuadores: Aplicaciones con Arduino.* Ciudad de México, México: Grupo Editorial Patria.

Enriquez Herrador, R. (2009). *Guia de Usuario de Arduino.* San Francisco, California: Creative Commons.

Freedom-Defined. (17 de 07 de 2017). Obtenido de http://freedomdefined.org/OSHW

Fry, B., & Reas, C. (08 de 2017). *Processing*. Obtenido de https://processing.org/

Garrido Abenza, P. (2015). *Comenzando a programar con Java.* Elche: Universitas Miguel Hernandez.

Guasco, I. (2012). *Tecnicas de Fotografia Profesional.* Buenos Aires, Argentina: RedUSERS.

INEGI. (14 de Marzo de 2016). *Instituto Nacional de Estadística y Geografía*. Obtenido de http://www.inegi.org.mx/saladeprensa/boletines/2016/especiales/especiales2016_03_01.pdf

INEGI. (07 de 2017). *TIC´s en hogares*. Obtenido de http://www.beta.inegi.org.mx/temas/ticshogares/

ISLab. (2017). *ISLab Intelligent Systems*. Obtenido de ISLab Intelligent Systems: http://islab.ulsan.ac.kr/research.php

Kang, Z. (29 de 04 de 2015). *Mesh Reconstruction from Imagery*. Obtenido de http://zhuoliang.me/meshrecon.html

Martin A. Fischler, O. F. (2014). *Readings in Computer Vision: Issues, Problem, Principles and Paradigms.* California: Morgan Kaufmann Publishers, Inc.

Mesa Múnera, E., Ramírez Salazar, J., & Branch Bedoya, J. (2010). Construcción de un modelo digital 3D de piezas precolombinas utilizando escaneo láser. *Avances en sistemas e informática.*, 197-206.

Orozco Quinceno, J. A., Romero Acero, Á., Marín Cano, A., & Jímenez Builes, J. A. (2014). Modelado 3D de objetos usando Matlab mediante sensor ultrasonico. *Revista Colombiana de Tecnologías de Avanzada*, 54-60.

Prometec. (15 de 08 de 2017). Obtenido de Prometec: http://www.prometec.net/motor-28byj-48/

Somolinos Sánchez, J. A. (2002). *Avances en robótica y visión por computador.* Castilla, España: Universidad de Castilla-La Mancha.

Suares y Alonso, R. C. (2010). *Tecnologías de la información y la comunicación.* Ideaspropias Editorial S.L.

Torrente Artero, Ó. (2013). *Arduino: Curso práctico de formación.* Madrid, España: RC Libros.

Vázquez, A. (21 de 04 de 2016). *Led&Go*. Obtenido de Led&Go: http://ledandgo.com/la-frecuencia-de-refresco-en-las-pantallas.html

Vilá Ubieto, K., Arranz Domingo, Á., Alvar Miró, M., & Sánchez Miralles, Á. (2009). *RECONSTRUCCIÓN 3D DE MODELOS UTILIZANDO TÉCNICAS DE VISIÓN ARTIFICIAL.* Madrid, España: UNIVERSIDAD PONTIFICIA COMILLAS.

Wu, C. (2014). *VisualSFM : A Visual Structure from Motion System*. Obtenido de http://ccwu.me/vsfm/