Implementación del Proceso de Manufactura Flexible de un Mini Implante en la Aleación de Titanio Ti – 6Al – 4V

Juan David RamírezVásquez

Universidad Santo Tomás.

Facultad de Ingeniería Mecánica

Bogotá D.C., Colombia

2017

Implementación del Proceso de Manufactura Flexible de un Mini Implante en la Aleación de Titanio Ti – 6Al – 4V

Juan David RamírezVásquez

Tesis o trabajo de investigaciónpresentada(o) como requisito para optar al título de:

Ingeniero Mecánico

Director:

DoctorJorge Andrés García Barbosa

Codirector:
Magister Oscar Rodrigo Lopez Vaca

Línea de Investigación:

Procesos de Manufactura

Grupo de Investigación:

Grupo de Estudios y Aplicaciones en Ingeniería Mecánica (GEAMEC)

Universidad Santo Tomás

Facultad de Ingeniería Mecánica

Bogotá D.C., Colombia

2017

Dedicatoria

A mi madre que me apoyo fielmente en todas las fases de mi formación académica como Ingeniero Mecánico.

Agradecimientos

El autor de este trabajo,agradece a la Universidad Santo Tomás, por la formación integral como Ingeniero Mecánico que ha recibido. También, a FODEIN por su asistencia económica para la realización del proyecto de investigación. Además, quiere agradecer a su director de trabajo de grado, el Doctor en Ingeniería, Jorge Andrés García Barbosa, por su guía en el desarrollo del trabajo de grado, del mismo modo, por todos los conocimientos y enseñanzas dadas a lo largo de mis estudios académicos de pregrado. Igualmente, agradezco, al candidato a Doctor en Ingeniería, Oscar Rodrigo López Vaca y al Ingeniero, John Alexander Zambrano; por su colaboración y efectiva complemento en la investigación descrita en este documento.

A mi madre, por su buena compañía, oraciones y deseos, ya que sin ellos no habría sido posible mi formación como profesional. Su constante apoyo en la hora de momentos de gran dificultad y sus buenos consejos. Por siempre creer en mí, en mis capacidades y exhortarme a seguir adelante sin importan las condiciones que se presentaran. A mi familia, por estar pendiente de mi a lo largo de la carrera.

A Edna Lucia Mogollón Montoya, por su especial amistad incondicional, por las incontables horas de escucha y consejos. A la familia Mogollón Montoya, por su gran recibimiento como un integrante más de su familia. A María Oveida Mogollón, por su preocupación sin interés. A todos ellos agradezco por ser una guía e inspiración para salir adelante.

Por último, pero no menos importante, atodas aquellas personas que de una u otra forma pusieron un granito para lograr mi meta de ser un profesional.

Resumen

Para el movimiento de dientes en ortodoncia, son utilizados los mini implantes dentales, representando un avance en este tipo de procedimientos. La tasa de falla de estos dispositivos es de entre el 13.5% - 16.4% debido a factores biológicos, mecánicos y ambientales. El análisis y diseño personalizado de mini implantes, reduciría la tasa de falla, pero implica retos en la eficaz elaboración de modelos tridimensionales y en su posterior proceso de manufactura.

En este proyecto se diseñó una aplicación embebida en un software de dibujo, para el modelamiento tridimensional paramétrico, que con las características geométricas, es capaz de modelar el mini implante personalizado de cada paciente.Además, se implementó un proceso de manufactura flexible para la fabricación de mini-implantes en aleación de titanio Ti – 6Al – 4V, de diferentes dimensiones, personalizados de acuerdo a las características biológicas de cada paciente. También, un lote muestra fue sometido a inspección de metrología para garantizar un producto de calidad. Los resultados son la automatización de un entorno de dibujo CAD, un proceso de manufactura con altos niveles de flexibilidad y de volúmenes de producción como un proceso en serie, y la participación a través de ponencia y poster, en el Congreso Internacional de Ingeniería Mecánica y Mecatrónica, realizado en la ciudad de Medellín en el 2017.

Palabras clave: Mini implante, API, Automatización CAD, Flexibilidad, Programación paramétrica, Micro torneado.

Abstract

For the teeth movement in orthodontics, mini dental implants are used, representing and advance in this type of procedures. The failure rate of these devices is between 13.5%-16.4% due to biological, mechanical and environmental factors. The analysis and custom

design of mini implants, would reduce the failure rate, but implies challenges in the efficient elaboration of three dimensional models and in their posterior manufacturing process.

In this project an application embedded in drawing software was designed for the three dimensional parametric modeling, which with the geometrical characteristics, is able to model the customized mini implant for each patient. In addition, a flexible manufacturing process was implemented for the manufacture of titanium Ti-6Al-4V alloy mini implants, of different dimensions, customized according to the biological characteristics of each patient. Also, a sample lot was subjected to metrology inspection to ensure a quality product. The results are the automation of a CAD drawing environment, a manufacturing process with high levels of flexibility, but also with production volumes of a serial process and participation through presentation and poster, at the International Congress of Mechanical Engineering and Mechatronics, in the Medellín city in 2017.

Keywords: Mini Implant, API, Customization CAD, Flexibility, Parametric programming, Micro turning.

Contenido

Pág.

Resumen ..... **IX**

Lista de figuras ..... **XIII**

Lista de tablas ..... **XV**

Lista de abreviaturas ..... **XVI**

1. Introducción **17**

2. Justificación **18**

3. Objetivos **19**

4. Marco Teórico **20**
4.1 Programación 20
4.2 Manufactura Flexible 22
4.3 Micro - Maquinado 24
4.4 Mini implantes 25
4.5 Programación Paramétrica CAD/CAM 26

5. Modelo Tridimensional Paramétrico **29**
5.1 Elaboración del modelo tridimensional de forma convencional 29
5.2 Automatización del proceso de dibujo. 35
5.2.1 Comunicación Librerías *Software* CAD – Visual Studio 37
5.2.2 Programación en Visual Studio 38
5.2.3 Comunicación API *Software* CAD 53

6. Plan de Manufactura **57**
6.1 Análisis de geometría, Operaciones tecnológicas y elección de maquinaria 57
6.2 Hoja de proceso 60
6.3 Código G, programación paramétrica 62
6.4 Código G, Programación con subprogramas 67

7. Montaje, fabricación y metrología **73**
7.1 Montaje y setup de la maquina 73
7.2 Fabricación 76
7.3 Metrología y compensación de herramientas 78

8. Conclusiones y recomendaciones **83**

8.1 Conclusiones 83
8.2 Recomendaciones 84

Bibliografía 85

Lista de figuras

Pág.

Figura 5-1:Geometría del mini implante. 31
Figura 5-2:Orden de construcción operaciones base 33
Figura 5-3:Operación de revolución para la generación de mini implante 34
Figura 5-4: Esquema de montaje para análisis por FEM 35
Figura 5-5:Estructura de actividades para la parametrización del modelo tridimensional 37
Figura 5-6: Adición de librerías al proyecto en Visual Studio 38
Figura 5-7: Conjunto Implante - Hueso - Platina 48
Figura 5-8:Interfaz gráfica API 52
Figura 5-9: Creación botón y enlace con API – 1 54
Figura 5-10:Creación botón y enlace con API – 2 55
Figura 5-11:Api adecuada al entorno de Solid Edge 55
Figura 6-1:Geometría mini implante dental 58
Figura 6-2:Herramientas adquiridas. Izq, pinzas de sujeción de materia prima de ¼". Der, inserto de 1.5 para tronzado ref 59
Figura 6-3: Aplicativo generación código de producción. 71
Figura 6-4: Metodología de implementación de un proceso de manufactura flexible y continuo. D: Diámetro, L: Longitud, LNR: Longitud no roscada, P: Paso. 72
Figura 7-1:Instalación pinza de sujeción y Mecanismo de ajuste presión pinza 74
Figura 7-2:Buje de sujeción. Izq vista posterior, Der vista anterior 74
Figura 7-3:Montaje herramienta de fresado 75
Figura 7-4:Procedimiento para la adecuación de recibimiento de información 77
Figura 7-5:Procedimiento de selección programa de producción 78
Figura 7-6:Lote de mini implantes fabricados. 78
Figura 7-7: Proyector de perfiles Mitutoyo PH350 79
Figura 7-8:Plano de metrología 80
Figura 7-9:Procedimiento compensación herramental 81

Lista de tablas

Pág.

Tabla 5-1: Caracterización mini implantes comerciales y de literatura..... 30

Tabla 5-2: Referencias y métodos para adición de entidades geométricas. 41

Tabla 5-3: Opciones de ProfilePlaneSide..... 44

Tabla 5-4: Métodos desarrollados..... 49

Tabla 6-1: Herramental utilizado en el mecanizado del mini implante 59

Tabla 6-2: Códigos G/M utilizados 63

Tabla 7-1: Datos registrados de la metrología a un lote de 3 mini implantes..... 80

Lista de abreviaturas

Abreviatura	Término
POO	Programación Orientada a Objetos
API	Aplication Programming Interface
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
IDE	Integrated Development Environment
RAM	Random Access Memory
SDK	Software Development Kit

1. Introducción

La permanente globalización junto con la necesidad de mejora de la competitividad y calidad en las empresas, al igual que el aumento de la demanda de productos personalizados y de una vida útil menor, han cuestionado los beneficios que posee el tradicional sistema de manufactura en serie para la producción de estos [1]. Según Cox (1986), manufactura flexible es la rapidez y facilidad de las plantas para adaptarse a las condiciones del mercado [2]. Debido a su gran adaptabilidad la manufactura flexible se ha convertido en foco de investigación ya que puede modificarse rápidamente a cualquier requerimiento del mercado.

A nivel médico y más específicamente en odontología, por el tamaño que posee el aparato estomatognático la mayoría de componentes mecánicos que se utilizan son de pequeñas dimensiones (menores a 1000 micras). El micro – maquinado por arranque de viruta se encarga de la fabricación de componentes de pequeñas dimensiones y representa un gran reto debido a las altas velocidades de corte que se deben alcanzar para obtener una buena calidad superficial en el producto [3].

Actualmente en ortodoncia entre el 13.5% - 16.4% de los implantes estandarizados fallan en su anclaje, lo cual es una tasa de falla alta; esto es debido a que las características biológicas de cada paciente y de cada lugar de anclaje son diferentes. Las técnicas y las características del lugar de anclaje son igualmente importantes como las características geométricas del implante es de esto que se propone personalizar la geometría de cada implante dependiendo del paciente y del lugar de anclaje con el fin de obtener un mejor resultado en el tratamiento [4].

Este proyecto pretende desarrollar un proceso de manufactura flexible de la parte roscada de un mini implante utilizado en ortodoncia; este componente se manufactura utilizando operaciones tecnológicas de micro torneado.

2. Justificación

En la industria médica es común hablar de productos personalizados debido a las características biológicas de cada paciente. Este nivel de personalización se alcanza teniendo procesos muy flexibles en donde se aplica manufactura a la medida, esto reduce los niveles de desperdicios de material, como también los niveles de stock, ya que se fabrica sobre pedido.

En consecuencia, la industria a través de la historia ha sufrido cambios de paradigma que comúnmente son llamados “revoluciones industriales”, la próxima revolución a la que nos enfrentamos es la 4.0, donde pasamos del paradigma de las Tecnologías de la Información para pasar al paradigma de las *Smart Industry’s*. El objetivo de este nuevo tipo de industria, es tener empresas que puedan ser capaces de adaptarse rápidamente a los diferentes cambios en el mercado, como también optimizar al máximo todo tipo de recursos requeridos en el proceso [5].

El internet, la cibernética, la necesidad de personalización y la adaptabilidad son los pilares que hacen posible este ideal generado en Alemania como un proyecto de investigación y que ahora se perfila para ser el modelo mundial de las industrias [5].

Esta investigación al plantearse el reto de generar un proceso de manufactura flexible para la fabricación de mini implantes, donde existen diferentes configuraciones para el mismo producto va en dirección con los objetivos de la industria 4.0. Además, busca la aplicación de la filosofía de Manufactura a la Medida con cero desperdicio y bajos niveles de stocks generando una base para la aplicabilidad de este tipo de industria en el país como también un incentivo en la investigación de este tipo de temáticas.

3. Objetivos

A continuación se muestran los objetivos de la investigación, compuestos por un objetivo general y tres objetivos específicos

Objetivo General

Implementar el proceso de manufactura flexible, para el tallado de la zona roscada, de un mini implante en la aleación de Titanio Ti-6Al-4V

Objetivos Específicos

Desarrollar un modelo tridimensional paramétrico, en la cual todas sus características geométricas se encuentren expresadas a través de relaciones matemáticas o geométricas.

Planificar un proceso de manufactura flexible para la parte roscada de un mini implante.

Fabricar un lote de mini implantes dentales con características geométricas diferentes.

4. Marco Teórico

A continuación, se definen los conceptos y temáticas claves para una buenacomprensión de los temas descritos más adelante en el documento.

4.1 Programación

- Método o Función

Los métodos o funciones, desde el punto de vista de programación, son vistos como una estrategia para la aplicación de la técnica: *divide y vencerás*. Un método es un conjunto de líneas de código, que en conjunto desempeñan una labor específica, ayudando en el desarrollo de un objetivo más grande. Estas, pueden ser utilizadas en diferentes partes del código pero las instrucciones que la definen solo se escriben una vez[6].

Una función se utiliza cuando es invocada dentro del programa principal, el llamado se realiza mediante el nombre e información requerida para la ejecución de su labor; a esta información se le denomina argumentos. Después del llamado, el método realiza la tarea para la cual fue diseñada y retorna al programa principal[6]

- Clase

Una clase es definida como el agrupamiento de variables y métodos que manipulan dichas variables[7]. Se puede definir una clase, como una guía / plano para la ejecución o elaboración de algo, y que además puede ser reutilizada múltiplesveces, siempre y cuando, ese algo a elaborar, sea del mismo tipo. Haga de cuenta que una clase, es como los planos de construcción de una casa, que contiene unas características y que describe

unos métodos para construir, pues bien, este plano no solo sirve para edificar esta casa, sino que también puede ser utilizado para construir cuantas casas se quieran [8].

En programación, una clase es utilizada para crear objetos, y a través de ellos acceder a las variables y métodos empaquetados en esta. Al interior de una clase existen 3 partes. La primera, ***private***, están todas las variables y métodos declarados bajo este esquema y que solo pueden ser accedidos desde el interior de la clase. La segunda, ***public***, que son todas las variables y métodos que pueden ser utilizados por otros, fuera de la clase. La tercera, ***protected***, identifica a aquellas variables y métodos que no podrán ser reconocidas por clases que deriven de la clase base[7].

- Objeto

Los objetos, en programación, son la instancia de una clase. Es decir, están representados por las variables y por los métodos declarados en la clase. Solo a través de ellos se podrán acceder a los métodos miembro. Se pueden crear cuantos objetos sean necesarios de la misma clase, de manera que para cada objeto existirán las variables declaradas como ***public*** o ***private***, lo cual también aplica para los métodos[6].

- Kernel

Es la parte fundamental que constituye a todo *software* y sistema operativo, siempre está en ejecución por debajo de las capas accesibles. También, puede constituirse como el conjunto de librerías, funciones o códigos que contienen la matemática principal de todo sistema informático.

- API's

Las API's por sus siglas en inglés (Aplication programming Interface), son pequeños aplicativos que sirven de comunicación con otros *software*. Por lo general son utilizadas para integrar datos y funcionalidades en diversos sistemas computacionales o informáticos. Actualmente, tienen un gran impacto por la gestión del rendimiento de los ***software's*** a los cuales pertenecen, como capas

abstractas (embebidos) [10]. Por lo general son utilizadas para aumentar las funcionalidades del *software*, a través de la comunicación con librerías abiertas de su *software* nativo.

4.2 Manufactura Flexible

La globalización de los mercados ha generado que los procesos se flexibilicen[11]. Actualmente muchas industrias han sido clasificadas como flexibles y aunque poseen un cierto nivel de automatización, realmente no poseen “Flexible Manufacturing Systems” (FMS); lo importante en un FMS es su nivel de automatización y la diversidad de partes que produce. Existen 8 tipos de flexibilidad según[12], a continuación se describe brevemente cada una de ellas.

- Flexibilidad en la Maquina: Se refiere a la facilidad con que se pueden cambiar el *setup* de la máquina para diferentes tipos de piezas, en ella viene involucrado el tiempo de cambio de herramientas desgastadas o en mal estado, el tiempo de cambio de las herramientas en el magazín y el tiempo de cambio de los diferentes sistemas de sujeción.

- Flexibilidad en el Proceso: Es la flexibilidad en poder fabricar piezas en diferentes tipos de materiales, la flexibilidad en el proceso es inversamente proporcional al costo de cambiar el setup de la máquina. En este tipo de flexibilidad cada pieza puede ser producida individualmente y no necesariamente en lotes.

- Flexibilidad en el Producto: Se trata de la rapidez y facilidad con la que un proceso puede pasar a producir un producto totalmente nuevo, también se puede tratar de la rapidez con la que cambio la producción a otro producto dentro de un conjunto de productos ya determinados.

- Flexibilidad en la Ruta de Producción: Es la baja disminución en los niveles de producción cuando se produce una avería en alguna maquina o línea del proceso. Esto se puede dar bien sea porque la pieza se puede producir en diferentes

líneas o porque las operaciones tecnológicas se pueden realizar en más de una máquina.

- Flexibilidad en el Volumen de Producción: Es la rentabilidad a la variación de los volúmenes de producción, esto se puede conseguir bajando los costos en el setup de la maquinaria y disminuyendo los costos por mano de obra directa. Esta flexibilidad es medida con cuán pequeños pueden ser los niveles de producción y que aun siga siendo rentable.

- Flexibilidad a la Expansión: La capacidad de tener un proceso y poderlo ampliar según las condiciones del mercado. Puede ser medido en cuán grande puede llegar a ser un FMS.

- Flexibilidad en la Operación: Si bien el orden de las operaciones en un tipo de pieza ya viene determinado por un orden jerárquico, existen otro tipo de operaciones las cuales no tiene dependencia. Por tal motivo es conveniente dejar este tipo de operaciones que no tienen dependencia abiertas para cualquier decisión que se pueda tomar en tiempo real.

- Flexibilidad de Planta: Es la mayor cantidad de diferentes tipos de partes que se pueden producir, viene dado por el nivel de tecnología que posee la fábrica. Se mide con el nivel de versatilidad y tecnología de las máquinas herramientas; se requiere tener todas las anteriores flexibilidades.

La flexibilidad en los procesos de manufactura se ha formado como un factor muy importante para la industria, sin embargo como se mencionó anteriormente esta no solo incluye el nivel de automatización. Esto exige una visión más amplia de lo que significa un FMS no solo en los diferentes niveles de flexibilidad o en los objetos de una fábrica sino que incluye todo un sistema en conjunto que debe adecuarse a la mutabilidad que pueda surgir en el nuevo mercado globalizado [11].

- Familia de partes

El concepto de familia de partes es utilizado por la filosofía de tecnología de grupos, en una búsqueda por la mayor eficiencia y calidad de los procesos de manufactura. La idea consiste en buscar similitudes en los componentes, para poderlos agrupar, de tal manera que sea más fácil su procesamiento. Una familia de piezas es un conjunto de partes que comparten características geométricas, en forma más no necesariamente en tamaño, o que simplemente poseen etapas de proceso similares en su fabricación [9].

4.3 Micro - Maquinado

El micro maquinado es un proceso de remoción de material para la producción de piezas con características geométricas menores a 500 micras. Inicialmente fue implementado para campos como la electrónica, biomecánica y aeroespacial pero poco a poco se ha venido aumentando la fabricación de micro piezas a tal punto que hoy en día ya se dispone en el mercado de herramientas con diámetros menores de 1 mm y hasta 0.05 mm tanto para fresado como para taladrado [13].

En micro fresado, taladrado y torneado son muy similares en su teoría con sus colaterales de mayor escala, pero representan mayores dificultades en cuanto a sus condiciones de corte. Esto se puede evidenciar, en el caso del torneado con el número de rpm el cual viene dado por:

$$N = v \cdot 1000/\pi \cdot Do$$ [14]

Donde:

N es el número de revoluciones por minuto en rpm

$v$es la velocidad de corte en m/min

Does el diámetro de la materia prima en mm (Decrece a medida que se maquina)

Nótese que si se mantiene constante la velocidad de corte el diámetro es inversamente proporcional a las revoluciones, por lo cual si se disminuye el diámetro las revoluciones aumentan. En micro torneado se puede trabajar con diámetros de entre 1 mm y 0.05 mm

produciendo revoluciones entre 22000 rpm y 150000 rpm que representan un reto importante. Esta misma teoría es aplicable para taladrado y fresado dando como requerimiento el poseer husillos de alta velocidad (<15000 rpm).

En términos de acabado superficial fue demostrado en [15], que el avance contribuye significativamente al mejoramiento de esta en el micro fresado; a bajos valores de avance se obtiene una superficie muy suave pero ineficiente en términos de tiempo. La formación de rebaba representa un gran reto debido a que su tamaño y debido a las pequeñas dimensiones de las herramientas, puede llegar a ser mayor al tamaño de la herramienta o al de la operación realizada. Para la disminución de rebaba en [15]recomiendan optar por estrategias de mecanizado discordante.

A pesar de la disponibilidad de herramientas para este tipo de micro mecanizados, la relación de avance por filo de corte es considerablemente mayor que en mecanizados a mayor escala, debido a ello es de gran importancia la buena selección de las condiciones de corte. Cuando las condiciones de corte son mal seleccionadas, las herramientas por su pequeño tamaño (D<0.5mm) pueden ser fácilmente quebradas y difícilmente inspeccionadas a simple visión de un operario en búsqueda de desgaste o falla [13].

4.4 Mini implantes

En la práctica de la ortodoncia se es necesario realizar movimientos en los dientes y para lograr ello se generan cargas sobre los dientes, este tipo de movimientos comúnmente son generados con ayuda de otros dientes o de accesorios. Estos métodos no siempre dan el resultado esperado creando limitaciones en el método pues puede producirse un movimiento indeseado en el diente ancla y en el caso de los accesorios, estos, producen gran malestar en los pacientes los cuales terminan por no utilizarlos [16].

Para 1945 Gainsforth and Higley, realizaron el primer anclaje de un mini implante con el fin de encontrar una alternativa a los métodos tradicionales de anclaje y tratando de solucionar el movimiento indeseado del diente ancla, este estudio fue realizado con 6 perros utilizando un mini implante de 3.6 mm de diámetro y 13 mm de longitud. Desafortunadamente los implantes se caían de entre 16 a 31 días después del anclaje [16].

La primera documentación que se tiene de un procedimiento de anclaje temporal es de 1983 realizada por Creekmore y Eklund, aunque esta práctica no fue inmediatamente aceptada por la falta de información acerca del tema y por el temor a complicaciones médicas. A partir de 1969 con el anclaje de un implante de puente realizado por Linkow ha sido bien documentado el anclaje de implantes temporales y su utilización en ortodoncia es cada vez mayor [16].

Los mini implantes son fabricados de aleación de Titanio Ti-4Al-6V [17] por su alta resistencia a la corrosión, por su interacción con células y proteínas, y por su alta curación en el hueso [18]. Estos dispositivos deben cumplir con características como: simple usabilidad, bajo costo, que pueda ser cargado inmediatamente después de su anclaje, dimensiones geométricas pequeñas, resistencia a cargas del tratamiento de ortodoncia, biocompatibilidad y que genere mejores resultados que los sistemas de anclaje tradicionales [16].

Actualmente sus dimensiones son muy variables y pueden ser descritos en la literatura con un diámetro de entre 1mm - 2mm y una longitud de entre 3mm - 12mm con roscas auto roscantes (algunos auto perforantes) de diferentes pasos y perfiles [16], [17], [19].

4.5 Programación Paramétrica CAD/CAM

La programación paramétrica es una herramienta desarrollada para aumentar la flexibilidad en los procesos de manufactura [20], en el desarrollo de modelos tridimensionales en *software* CAD y en el mayor envío de información entre CAD - CAM. Este tipo de programación en el campo CAM, no es más que códigos G/M en donde son especificadas las posiciones de cada uno de los ejes, el avance y las revoluciones, como variables que poseen un tipo y un valor [21].

En el Control Numérico Computarizado (CNC) se generan programas NC, que si bien, no poseen una programación avanzada, si es posible la realización de códigos de mayor complejidad a través de programación paramétrica, para simplificar líneas de código. Además, para partes con geometrías similares (familia de partes), es posible la creación de macros que parametricen los códigos G, en función de características especiales;

también, es aplicable para piezas que aunque no tienen similitud en el diseño si requieren de operaciones tecnológicas similares [21].

A nivel CAD se habla de los niveles de flexibilidad del modelo, y se busca que los modelos posean la mayor flexibilidad y adaptabilidad a cambios inesperados en el diseño, también se buscala reutilización de algunas características para nuevos modelos. Sistemas CAD paramétricos facilitan la creación de modelos con un alto nivel de adaptabilidad en donde ciertas características están definidas en base a parámetros, lo que permite diferentes configuraciones del mismo producto con solo cambiar algunos valores [22].

5. Modelo Tridimensional Paramétrico

El anclaje de los implantes dentales es dependiente de los factores biológicos del paciente, como también, de las condiciones de anclaje. Teniendo como resultado, que las características geométricas del implantepara cada paciente, sean diferentes. En términos de modelamiento, esta variabilidad de los mini implantes, resulta un reto el cual será abordado, a través de parametrización.

El modelamiento paramétrico descrito aquí, es una herramienta disponible para la mayoría de *software's* CAD en la cual se permite acceder al *Kernel* y utilizarlo para desarrollar modelamientos desde un nivel más básico, permitiendo obtener una mayor personalización en los modelamientos. En esta ocasión, el acceso al *Kernel* se realiza a través de Visual Basic y el *Kernel* utilizado es propiedad de Solid Edge.

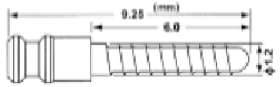
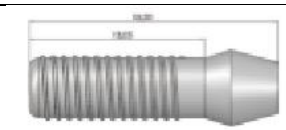
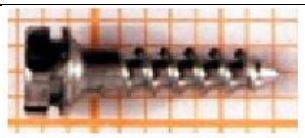
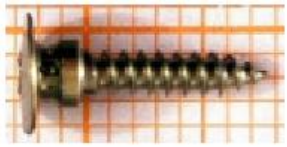
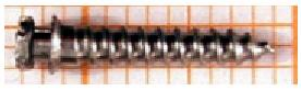
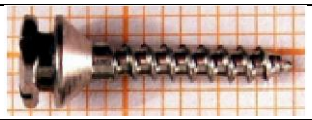
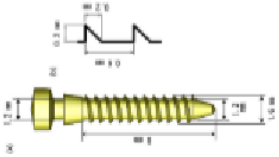
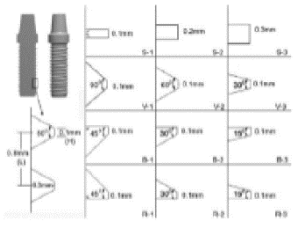
Inicialmente, encontrará el análisis a partir del cual se obtuvieron los pasos lógicos para la generación del modelo tridimensional de un mini implante, sin importar sus características geométricas. Posteriormente, se evidencia el diseño de clases para la elaboración del código fuente utilizando la filosofía de Programación Orientada a Objetos (POO). Como último, se muestran las labores realizadas para la comunicación entre Visual Basic y Solid Edge, y también se hace una breve descripción de las funciones y algoritmos del *Kernel* utilizados para el modelamiento paramétrico del mini implante.

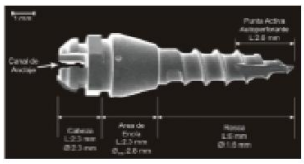
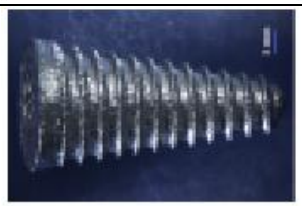
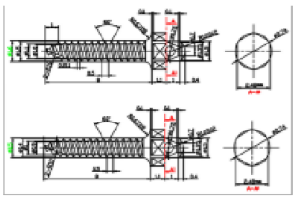
5.1 Elaboración del modelo tridimensional de forma convencional

En el diseño de mini implantes dentales, la geometría ha sido modificada múltiples veces a lo largo de los años, pero ya está generalizada una geometría que arroja los mejores resultados (Estructura roscada sobre superficies cilíndricas o cónicas). Las variaciones de estos diseños están dadas por cambios en el diámetro, en la longitud, en el paso y

enel perfil de la rosca. En la Tabla 5-1 se presenta las principales características de algunos mini implantes comerciales y de la literatura.

Tabla 5-1: Caracterización mini implantes comerciales y de literatura

Referencia	Material	Longitud Cabeza	Longitud Roscada	Tipo de Rosca	Angulo Rosca	Diametro Exterior	Paso	Tipo de Eje	Imagen
Orthoanchor K1 system	Titanio puro	3.25	6	-	-	1.2	-	Cilindrico	
-	Titanio denso - Titanio poroso: 40, 50, 60, 70%	5.45	13.05	ACME	30	3.98	-	Cilindrico	
Aarthus	Aleación de titanio (ASTM F136)	1.7	7.9	Autoperforante	-	2	1	Cilindrico	
FAMI	Aleación de titanio (ASTM F136)	2	8	-	-	2	0.75	Conico	
Dual Top	Aleación de titanio (ASTM F136)	2	10	-	-	2	1	Conico	
Spider	Aleación de titanio (ASTM F136)	3.6	11	Autoperforante	-	2	1	Cilindrico	
-	Titanio	-	8	Diente de sierra	-	1.6	0.6	Cilindrico	
Biohorizon	Titanio	7	13	Cuadrada	-	3.9	0.8	Cilindrico	
Nobel Biocare, 3I	Titanio	7	13	En v	-	3.9	0.8	Cilindrico	
Steri - Oss	Titanio	7	13	Buttres	-	3.9	0.8	Cilindrico	
Steri - Oss, CA	Titanio	7	13	Reverse Buttres	-	3.9	0.8	Cilindrico	
-	Titanio (Ti - 6Al - 4V)	-	11.5	-	-	3.75	-	Cilindrico	-

-	Titanio (Ti - 6Al - 4V)	4.6	6	Mitad Autoperforante	-	1.6	-	Conico	
-	Titanio Grado 2	-	-	Triangular	-	-	1	Conico	
Dentos	Titanio	2.5	8	Triangular - Autoperforante	60	-	0.5	Cilindrico	

Adaptado de: Informe Final FODEIN 2016 del proyecto titulado: "Desarrollo de un dispositivo de anclaje temporal (DAT), para ortodoncia, en función de sus características geométricas y la densidad ósea del maxilar"

Estas modificaciones están determinadas por factores biológicos del paciente tales como la densidad ósea y las condiciones del lugar de anclaje. Tomando en cuenta que la densidad ósea de cada paciente es diferente, unido a las innumerables condiciones que se pueden presentar en el lugar de anclaje, da como resultado la necesidad de un análisis más personalizado para cada caso.

A consecuencia de lo anterior, el modelamiento tridimensional de mini implantes personalizados resultaría tedioso y consumiría gran cantidad de tiempo en la etapa de dibujo, debido a la constante repetición de operaciones de modelado. Si bien se presentan cambios en dimensión, los rasgos geométricos y la metodología del modelamiento es la misma, resultando de ello un proceso que puede ser automatizable.

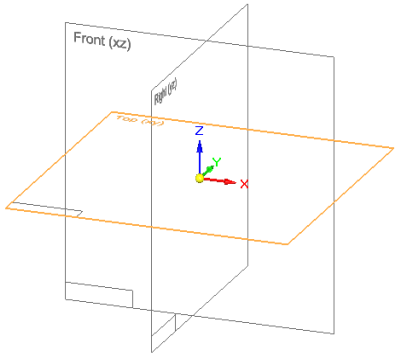
La geometría del mini implante a modelar se muestra en la Figura 5-1. Este es de fabricación brasileña de la empresa Conexao.

Figura 5-1: Geometría del mini implante.

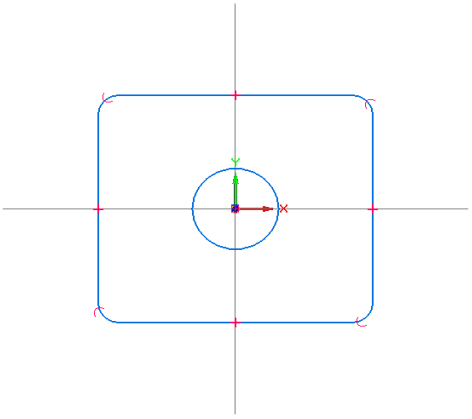
Para la automatización del proceso de modelamiento del mini implante, se debe conocer la manera convencional de hacer el procedimiento. El modelamiento en 3D posee algunas operaciones básicas para la generación de cualquier objeto, entre ellas, se puede destacar la extrusión, la revolución, el barrido y el corte. Los pasos para la ejecución de cada una de estas operaciones en Solid Edge,tienen un orden, que independientemente de la operación, es el mismo.

La operación inicia con la elección de un plano, que bien puede ser un plano de referencia o una de las caras ya dibujadas del objeto; sobre la cual se dibuja unboceto cerrado con sus respectivas dimensiones y relaciones geométricas. Una vez realizado este procedimiento, se aplica una de las operaciones básicas indicando el valor de la operación (en el caso de la extrusión, se deberá indicar la distancia a extrudir; para la revolución se dará el ángulo a revolucionar). En la Figura 5-2 se visualiza cada uno de los pasos descritos anteriormente, incluyendo dos de las cuatro operaciones básicas.

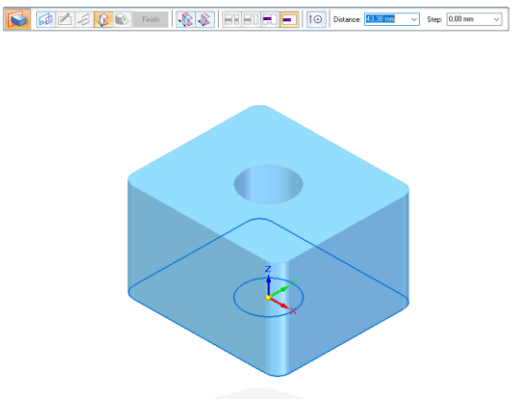
Figura 5-2: Orden de construcción operaciones base.



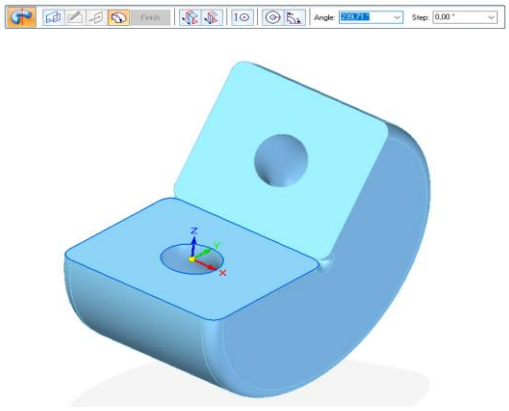
(a) Elección plano referencia.



(b) Bosquejo con geometría cerrada



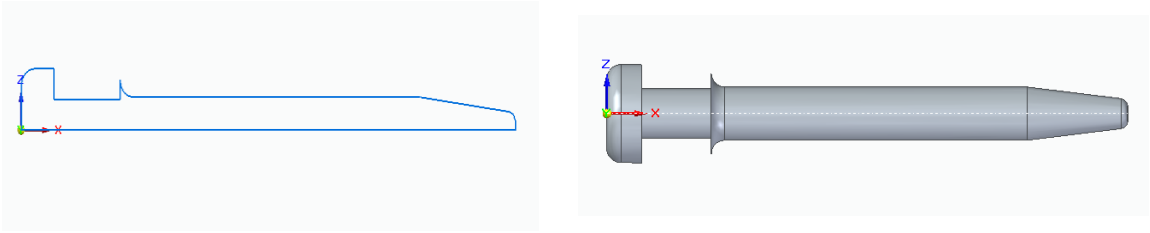
(c) Extrusión



(d) Revolución

El modelamiento del mini implante, inicia con la selección del plano de referencia x-y y sobre este se dibuja la geometría de la Figura 5-3a. Se realiza una revolución del bosquejo teniendo como eje la línea inferior paralela al eje x con un ángulo de 360°, dando como resultado lo visto en la Figura 5-3b. Para el hexágono, se crea un plano paralelo al plano de referencia x-z y a una distancia de 2 mm, en él se dibuja el bosquejo del hexágono y como último se realiza la extrusión de 1 mm.

Figura 5-3: Operación de revolución para la generación de mini implante



(a) Bosquejo revolución mini implante (b) Resultado operación de revolución

El cuerpo roscado se construye gracias a la herramienta de barrido en hélice. Esta herramienta requiere como todo barrido, de un perfil y de una trayectoria por la cual se extenderá. El perfil inicialmente trabajado para la rosca del mini implante fue métrico a 60°. Para el dibujo de la trayectoria se realizó un bosquejo 3D con la herramienta de hélice, proporcionando los valores de paso y longitud.

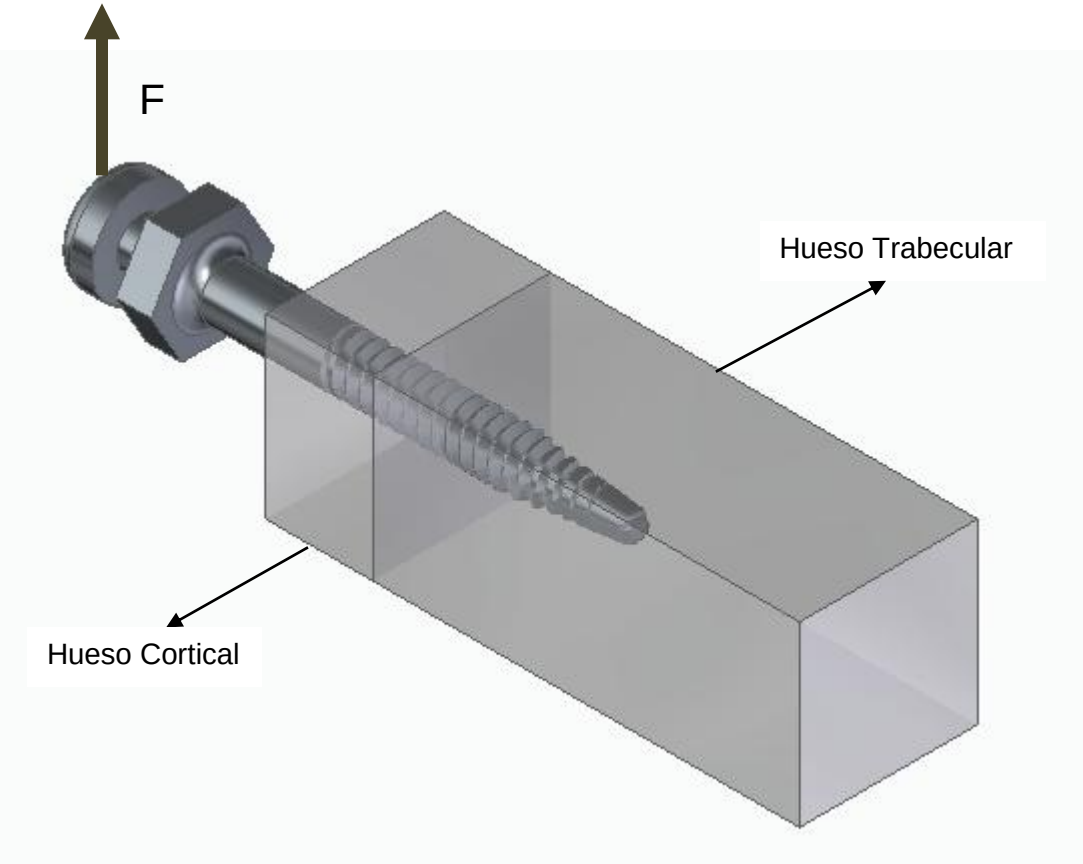
En el último paso del proceso, se deben realizar los cortes correspondientes a los filos del mini implante por medio de un corte en extrusión. El diseño de los filos corresponde a la geometría de las herramientas disponibles para su manufactura, y también de las teorías de corte para hueso.

Como se mencionó, los mini implantes tienen variaciones debido a las características biológicas de cada paciente, llevando a que en cada caso se realice un modelo tridimensionalpersonalizado. El tener un diseño paramétrico que tenga en cuenta las características de cada paciente, también implica tener un proceso de manufactura capaz de fabricar cada implante diseñado. Esta tarea, a través del método convencional implica mayor consumo de tiempos en la etapa de diseño, y en algunas industrias donde ya tienen arraigadas tecnologías CAM, en la etapa de manufactura. A su vez, esto redunda en mayores costos de producción que se ven reflejados en el valor final del producto; siendo el usuario el más afectado y pudiendo llegar a desincentivar su consumo.

Además, el modelamiento tridimensional paramétrico juega un rol importante en la etapa de diseño, que contempla análisis por elementos finitos, ya que permite de manera flexible, la generación de modelos para su posterior análisis FEM. En la Figura 5-4se muestra el esquema del montaje requerido, donde encontramos el implante anclado al hueso, diferenciando las capas de hueso, cortical y trabecular, cuyas características

mecánicas son diferentes, y una fuerza vertical aplicada en la cabeza del implante (Ver Anexo C).

Figura 5-4: Esquemademontaje para análisis por FEM



5.2 Automatización del proceso de dibujo.

Como se mencionó anteriormente, los altos costos de producción generan la necesidad de reducir los tiempos de modelamiento tridimensional,siendo una de las posibilidades, la reutilización de informaciónque es generada de forma repetitivadurante el proceso de modelado de mini implantes personalizados. Tal información se evidencia en valores de operaciones, bosquejos, operaciones de modelado y medidas no variantes en la geometría.

Una forma de reutilización de información en modelamiento tridimensional es la parametrización y dependiendo del tipo, se pueden obtener diferentes niveles de flexibilidad. La parametrización consiste en la creación de una tabla de variables donde

es posible la modificación del modelo (mas no la del perfil de rosca) y se aplican sobre el mismo archivo de manera ágil, pero la anterior versión a este se pierde. Por esa razón esta solución no es aplicable.

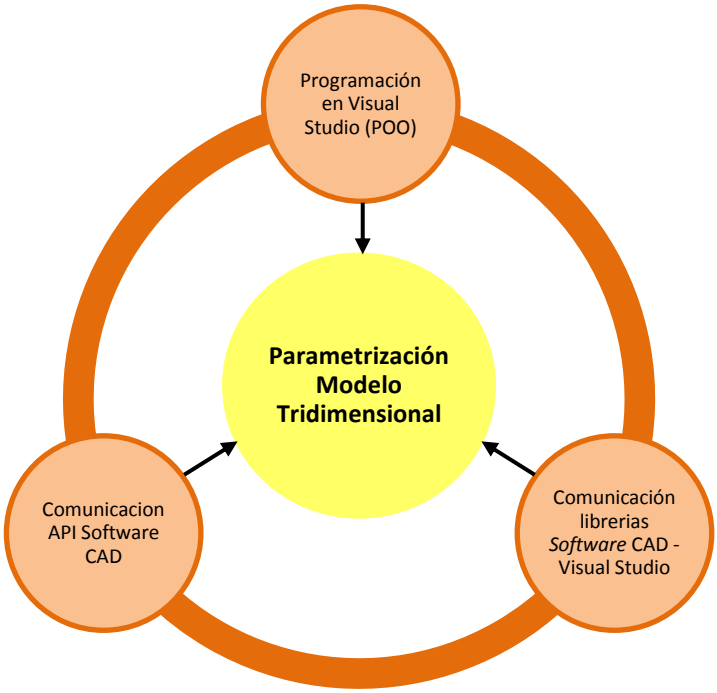
Por el contrario, la parametrización a través de programación, que por medio de herramientas del *Kernel* del *software* CAD permite la obtención de un mayor nivel de flexibilidad,parametrizando bosquejos e inclusive operaciones, logra que las diferentes versiones que se requieran del modelo puedan ser almacenadas en archivos distintos. El resultado de esta programación es una APIenlazada a un botón que se crea en el *software*.

Para la programación de la APIse deben tener en cuenta varias nociones. Primero, la programación se debe realizar desde un Entorno Integrado de Desarrollo (IDE) que pueda acceder a las librerías propias del *software*. Segundo, la programación aunque se puede realizar de manera estructurada se recomienda sea elaborada con el paradigma de la Programación Orientada a Objetos (POO), que facilita el entendimiento y posibles futuras modificaciones. Tercero, el resultado de la programación, es un ejecutable que debe ser comunicado con un botón previamente creado en el *software* CAD de lo contrario el ejecutable por sí solo no realizará su tarea.

A continuación en la Figura 5-5,se muestra la estructura de las actividades que deben realizarse para cumplir con el objetivo de parametrización del modelo tridimensional, a través de la programación embebida. Como se observa, las actividades están relacionadas y son dependientes entre sí, por tal motivo, se explicará cada una de ellas en el orden en que se deben ejecutar.

Todo el desarrollo de la parametrización del modelo tridimensional, se desarrolla en Visual Studio 2015 en una computadora con procesador Intel Core i5 y 8 Gb de memoria RAM. El *Software* CAD utilizado es Solid Edge versión 8.Al ser programación basada en los códigos raíz, el APIpodrá comunicarse con cualquier versión de Solid Edge.

Figura 5-5: Estructura de actividades para la parametrización del modelo tridimensional



5.2.1 Comunicación Librerías *Software* CAD – Visual Studio

El acceso a las clases y funciones del *Kernel* se realiza por medio de librerías de Solid Edge. En estas librerías vienen contenidas todas las clases, métodos y atributos requeridos para realizar la parametrización del modelo tridimensional y ya vienen incluidas en el IDE[1] – Visual Studio, pero deben ser añadidas al proyecto en cuestión.

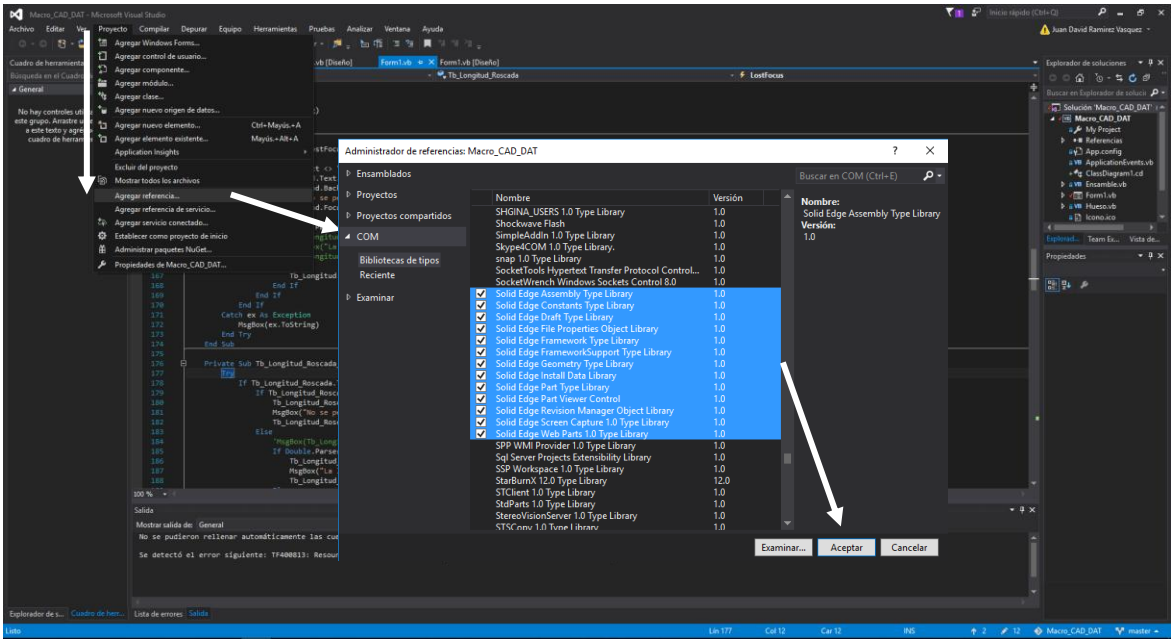
Para la adición de las librerías se debe acceder desde el IDE al desprendible Proyecto ubicado en la barra de herramientas y acceder a la ruta Proyecto>>Agregar referencia>>COM>>Biblioteca de tipos (ver Figura 5-6). Concluida esta labor, el proyecto se encuentra listo para la programación requerida. Aunque se adicionaron todas las librerías, cabe mencionar que no todas son utilizadas; las librerías que se requirieron para este proyecto son:

- Solid Edge Assembly Type Library

[1] Se incluyen automáticamente al momento de la instalación de Solid Edge.

- Solid Edge Constants Type Library
- Solid Edge FrameworkSupport Type Library
- Solid Edge Part Type Library

Figura 5-6: Adición de librerías al proyecto en Visual Studio



5.2.2 Programación en Visual Studio

A continuación se realiza una breve descripción de los métodos del *Kernel* utilizados en la parametrización del modelamiento tridimensional.Además, se describen cada uno de los métodos desarrollados por el autor, como estrategia para una mayor eficiencia en la escritura del código ysiendo parte fundamental para la implementación de la metodología POO utilizada en la programación. Es de tener en cuenta que todos los ejemplos **NO** son una copia fiel de lo escrito en los códigosfuente y que por el contrario, tienen pequeñas modificaciones para una mejor asimilación por parte del lector. En elAnexo Ade este documento, se encuentra el código fuente desarrollado.

- **Conexión con *Software* CAD y creación del archivo parte**

Lo primero que se realiza, es la conexión con el *software* Solid Edge, para ello se crean los siguientes 3 objetos:

```
Dim objApp As SolidEdgeFramework.Application = Nothing
Dim objDocuments As SolidEdgeFramework.Documents = Nothing
Dim objPart As SolidEdgePart.PartDocument = Nothing
objApp = Marshal.GetActiveObject("SolidEdge.Application")
objDocuments = objApp.Documents
objPart = objDocuments.Add("SolidEdge.PartDocument")
```

El objeto `objApp`inicialmente, verifica que Solid Edge se encuentre abierto para posteriormente enlazarse con él;`objDocuments`, sirve para crear un archivo tipo parte del cual, quedará almacenado en `objPart`.

- **Generación de plano**

La creación de plano, se realiza de dos maneras, creando una copia de alguno de los planos iniciales del *software* (Ver Figura 5-2), o en paralelo a una referencia (planos referencia, característica geométrica del modelo, plano o superficie). A continuación se exhibe el código para el primer método:

```
Dim Perfil as SolidEdgePart.Profile = Nothing
Perfil = objPart.ProfileSets.Add().Profiles.Add(RefPlanes.Item(NumPlane))
```

`objPart`es obtenido de la creación del archivo y nos indica que se está accediendo a los métodos de un archivo tipo parte. `ProfileSets.Add()`, permite la creación de la configuración para un nuevo plano. `Profiles.Add(Ref)`,es el método encargado de la creación del plano. Este método recibe un solo parámetro, `Ref`. En `Ref` debe ser indicado cómo se va a realizar la creación, esta puede ser por copia de una referencia, paralelo a esta, o en ángulo a esta. Para un plano por copia simplemente se indica la referencia `RefPlane.Item(NumPlane).`Donde `RefPlane`,es un arreglo que almacena los tres planos de referencia del *software* (xy, yz, xz) y `NumPlane` indica el número del plano al cual se quiere hacer referencia.

En el caso de querer un plano paralelo a una referencia, se utiliza la siguiente línea de código:

```
Dim Perfil asSolidEdgePart.Profile = Nothing
Perfil = _
       objPart.ProfileSets.Add().Profiles.Add(_
       RefPlanes.AddParallelByDistance(RefPlanes.Item(NumPlane),_
```

```
Dis, SolidEdgePart.ReferenceElementConstants.igNormalSide))
```

La diferencia de esta línea con la del anterior método, es que se debe indicar el método para la adición de un plano paralelo a una distancia (`AddParallelByDistance`), el cual recibe tres parámetros: la referencia (`RefPlanes.Item(NumPlane)`), la distancia de creación del plano paralelo (`Dis`) y una constante de condición geométrica especial (`SolidEdgePart.ReferenceElementConstants.igNormalSide`); para este ejemplo se utiliza `igNormalSide`, debido a que no se desea ninguna condición geométrica.

- **Construcción de bosquejos**

El siguiente paso es la generación del sketch a través de entidades geométricas básicas (puntos, líneas, círculos, arcos).Antes de la creación de cualquier línea hay que tener en cuenta que se requiere de un objeto tipo `SolidEdgePart.Profile`que es el plano donde se dibujará el bosquejo. Este plano, se debe crear basado en alguno de los métodos descritos en la viñeta Generación de plano.

Se crean dos objetos para la elaboración de cualquier entidad geométrica, el primer objeto accederá a la colección referencia y el segundo será el encargado de dibujar la entidad. En la Tabla 5-2se muestran las referencias y métodos de adición para cada una de las entidades geométricas y a continuación, se da un ejemplo para la construcción de una línea; tenga en cuenta que el objeto "`Perfil`" utilizado en este ejemplo no es más que un nombre y el mismo se deberá crear basado en el código descrito en la viñeta de Generación de plano:

```
Dim objeto1As SolidEdgeFrameworkSupport.Lines2d = Nothing
Dim objeto2 As SolidEdgeFrameworkSupport.Line2d = Nothing
objeto1 = Perfil.Lines2d
objeto2 = objeto1.AddBy2Points(x1, y1, x2, y2)
```

El método utilizado para añadir una línea es `AddBy2Points(x1, y1, x2, y2)` donde los parámetros recibidos, son las coordenadas de los puntos de inicio y finalización de la línea; estas coordenadas deben ser ingresadas en metros. Como se puede observar, ambos objetos pertenecen a la librería `SolidEdgeFrameworkSupport` en la cual se encuentran todos los recursos necesarios para la elaboración de entidades geométricas básicas. Observe que el `objeto1`, es un arreglo y que al adicionar una línea a través del `objeto2`,se crea un nuevo ítem por lo cual todas las adiciones que se hagan

quedaránalmacenadas en algún espacio de este. Nótese también, que el ejemplo dado se puede aplicar para cualquier entidad geométrica; solo se requiere de la modificación de las referencias y utilizar alguno de los métodos mostrados en la Tabla 5-2.

Tabla 5-2: Referencias y métodos para adición de entidades geométricas.

Entidad Geométrica	Referencias	Métodos para adicción
Arco	Arc2d	AddAsFillet
		AddAsFilletNoTrim
	Arcs2d	AddByCenterStartEnd
		AddByStartAlongEnd
Circulo	Circle2d	AddBy3Points
	Circles2d	AddByCenterRadius
Elipse	Ellipse2d	AddByCenter
	Ellipses2d	
Línea	Line2d	AddAsChamfer
		AddBy2Points
	Lines2d	AddByPointAngleLength
Punto	Point2d	Add
	Points2d	

- **Restricciones geométricas**

Este paso puede ser omitido, pero es recomendable su ejecución ya que se evitan conflictos de forma al momento de aplicar alguna de las operaciones de modelamiento. Las restricciones geométricas son un control para garantizar que los bosquejos tengan un perfil cerrado. Al igual que con la creación de las entidades geométricas, se deben crear dos objetos donde el primero accederá a la referencia y el segundo es el encargado de realizar la restricción. A continuación, se presenta un ejemplo de restricción y es de aclarar que no es laúnica que se puede aplicar:

```
Dim objeto1As SolidEdgeFrameworkSupport.Relations2d = Nothing
Dim objeto2As SolidEdgeFrameworkSupport.Relation2d = Nothing
```

```
objeto1 = Perfil.Relations2d
objeto2 = objeto1.AddKeypoint(Lines2d.Item(1), _
KeypointIndexConstants.igLineEnd, _
Lines2d.Item(2), _
              KeypointIndexConstants.igLineStart)
```

Tenga en cuenta que el objeto "`Perfil`", utilizado en el ejemplo, no es más que un nombre y el mismo se deberá crear basado en el código descrito en la viñeta de Generación de plano, y el objeto "`Lines2d`", hace referencia a `objeto1` creado en la viñeta deConstrucción de bosquejos.

La restricción ilustrada es un *KeyPoint*con la función de unir dos líneas por medio de sus puntos iniciales y finales. El `objeto1`, al igual que en los ejemplos anteriores, accede a las referencias y también es un arreglo que almacena lo creado. El `objeto2`, es una plataforma para la adición de restricciones a través de diferentes métodos, en este caso `AddKeypoint(L1,P1,L2,P2)`. Este método recibe cuatro parámetros, aquellos que inician por L (L1, L2) indican las líneas que se van a unir y aquellos que inician por P (P1, P2) hacen referencia a los puntos de inicio o final de cada línea por los cuales se unirán; estos puntos se indican mediante las `KeypointIndexConstants`, con los atributos `igLineEnd` para puntos finales y `igLineStart` para puntos iniciales.

Una vez realizadas todas las restricciones, se debe verificar que el perfil esté cerrado, para ello, se utiliza el método`End`sobre el perfil creado en la viñeta Generación de plano; si el perfil está abierto, el método retorna un valor diferente de cero de lo contrario, retornara 0. En seguida se encuentra un ejemplo de la línea de código utilizada.

```
Perfil.End(SolidEdgePart.ProfileValidationType.igProfileClosed)
```

- **Operaciones de modelado (Revolución, Extrusión, Barrido por hélice, Corte)**

Una vez generado el perfil con sus respectivas restricciones y validaciones, se aplica una operación de modelado. Para el proyecto se utilizan tres (Revolución, Extrusión, Barrido por hélice, Corte). Cada una de ellas es realizada por un método distinto, el cual requiere de objetos y parámetros creados anteriormente.

Revolución:En el caso de la Revolución para la elaboración del cuerpo roscado del implante, se efectuaron las siguientes líneas de código.

```
Dim objeto1 As SolidEdgePart.Models = Nothing
Dim objeto2 As SolidEdgePart.Model = Nothing

Dim objeto1 = Part.Models

Dim Array(0 To 1) As SolidEdgePart.Profile
Array(0) = Perfil

objeto2 = objeto1.AddFiniteRevolvedProtrusion(_
       1, _
       Array, _
       EjeRevolucion(9, Perfil), _
       SolidEdgePart.FeaturePropertyConstants.igRight,_
       Ang * Math.PI / 180)
```

Al igual que en los códigos explicados anteriormente, también se utilizan dos objetos, teniendo el primero para acceder a la referencia y el segundo, como encargado de realizar la operación a través del método:

```
Public Function AddFiniteRevolvedProtusion( _
   ByVal NumberOfProfiles As Long, _
   ByRef ProfileArray() As Object, _
   ByVal ReferenceAxis As RefAxis, _
   ByVal ProfilePlaneSide As FeaturePropertyConstants, _
   ByVal AngleofRevolution As Double, _
   Optional ByVal KeyPointOrTangentFace As Variant, _
   Optional ByVal KeyPointFlags As Variant _
) As Model
```

Donde, `NumberOfProfiles`es el número de perfiles que van a ser utilizados para crear la operación, `ProfileArray()`, es el arreglo donde estarán almacenados los bosquejos a utilizar. Es por ello que en el código de ejemplo se declara un arreglo unidimensional "`Array`" pero de tipo `SolidEdgePart.Profile`,con el fin de no tener incompatibilidad con los datos a almacenar en él.`ReferenceAxis`, como su nombre lo indica es el mismo eje de referencia utilizado por el método convencional sobre el cual se hace la revolución y el cual se configura de la siguiente manera.

```
Perfil.SetAxisOfRevolution(Perfil.Lines2d.Item(index))
```

Observe que en `Perfil.Lines2d.Item(index)`, se selecciona una de las líneas del perfil guardada en uno de los espacios del arreglo `Lines2d` (Index) y que el método `SetAxisOfRevolution`, la configura como un eje de referencia. Tenga en cuenta que las palabras "Perfil" y "Lines2d", deberán corresponder a los objetos creados en los apartados de Generación de plano y Construcción de bosquejos.

`ProfilePlaneSide`,estableceel sentido de revolución, hacia el lado negativo, positivo o en ambos lados con respecto al perfil; es una constante miembro `FeaturePropertyConstants` con tres posibilidades posibles mostradas en la Tabla 5-3. Por último,se debe indicar en `AngleofRevolution`, el ángulo de revolución medido en radianes.

Tabla 5-3: Opciones de ProfilePlaneSide

Característica	Opción	Código
Sentido del barrido en la operación	Lado negativo	`SolidEdgePart.FeaturePropertyConstants.igRight`
	Lado positivo	`SolidEdgePart.FeaturePropertyConstants.igLeft`
	Ambos lados	`SolidEdgePart.FeaturePropertyConstants.igBoth`

Extrusión:El código para realizar la extrusión del hexágono, de los emuladores de hueso corticaly hueso trabecular se presenta a continuación:

```
Dim objeto1 As SolidEdgePart.Models = Part.Models
Dim objeto2 As SolidEdgePart.Model = Nothing

Dim Array(0 To 1) As SolidEdgePart.Profile
Array(0) = Perfil

objeto2 = objeto1.AddFiniteExtrudedProtrusion(1,_
       Array,_
       SolidEdgePart.FeaturePropertyConstants.igRight,_
       Dis)
```

Se efectúa de forma similar a la revolución creando dos objetos con las funciones ya mencionadas en los anteriores ejemplos y valiéndose del siguiente método:

```
Public Function AddFiniteExtrudedProtrusion( _
   ByVal NumberOfProfiles As Long, _
   ByRef ProfileArray() As Object, _
   ByVal ProfilePlaneSide As FeaturePropertyConstants, _
   ByVal ExtrusionDistance As Double, _
   Optional ByVal KeyPointOrTangentFace As Variant, _
   Optional ByVal KeyPointFlags As Variant, _
   Optional ByVal FromSurfOrRefPlane As Variant, _
   Optional ByVal ToSurfOrRefPlane As Variant _
) As Model
```

Donde, `AddFiniteRevolvedProtusion`, es el número de perfiles que van a ser utilizados para crear la operación y `ProfileArray()`, es el arreglo donde estarán almacenados los bosquejos a utilizar. Es por ello que en el código de ejemplo se declara un arreglo unidimensional "`Array`" pero de tipo `SolidEdgePart.Profile`,con el fin de no tener incompatibilidad con los datos a almacenar en él.`ProfilePlaneSide`,representa el sentido de barrido en la extrusión, teniendo la posibilidad de ser positivo, negativo o en ambos sentidos; en la Tabla 5-3 se presentan las posibles características a utilizar. Por último, se indica la distancia a extrudir, medida en metros.

Barrido en hélice:En la operación de corte en barrido para la elaboración de la rosca, se requiere de una función que sea capaz de realizar un barrido en hélice y con ello evitar la elaboración de una trayectoria. Esto se logra debido a que el método mostrado a continuación construye implícitamente su trayectoria, a través de los parámetros indicados. Es de aclarar, que para implementar este código, se debe tener de antemano el perfil a barrer, que en este caso en particular, es el perfil de la rosca deseada.

```
Dim objeto1 As SolidEdgePart.HelixCutout = Nothing
Dim Array As Array = Nothing
Array = Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)

If TipoRosca Then
       Array.SetValue(PerfilMetrico(Part, Me.ProfileRosca), 0)
Else
       Array.SetValue(PerfilSierra(Part, Me.ProfileRosca), 0)
End If
```

```
objeto1 = Part.Models(0).HelixCutouts.AddFinite(EjeRevolucion(4,_
       Me.ProfileRosca),_
       SolidEdgePart.FeaturePropertyConstants.igStart,_
       1,_
       Array,_
       SolidEdgePart.FeaturePropertyConstants.igRight,_
       0.1,_
       Me.Paso,_
       Me.LongRos / Me.Paso,_
       SolidEdgePart.FeaturePropertyConstants.igRight)
```

En esta ocasión, solo se crea el `objeto1`, ya que el segundo objeto se encuentra implícito en `Part.Models(0).HelixCutouts`, donde `Part` hace referencia al objeto creado en el apartado Conexión con *Software* CAD y creación del archivo parte. También, al igual que en las anteriores operaciones de modelado,se crea un arreglo para almacenar los perfiles que componen la operación, pero al contrario de lo visto hasta ahora, se diseña un control de decisión para que dependiendo de la selección del usuario, se utilice el perfil de rosca seleccionado. El barrido en hélice se adiciona al modelo, mediante el siguiente método.

```
Public Function AddFinite( _
   ByVal HelixAxis As RefAxis, _
   ByVal AxisStart As FeaturePropertyConstants, _
   ByVal NumCrossSections As Long, _
   ByRef CrossSectionArray() As Object, _
   ByVal ProfileSide As FeaturePropertyConstants, _
   ByVal Height As Double, _
   ByVal Pitch As Double, _
   ByVal NumberOfTurns As Double, _
   ByVal HelixDir As FeaturePropertyConstants, _
   Optional ByVal TaperAngle As Variant, _
   Optional ByVal TaperSide As Variant, _
   Optional ByVal EndPitch As Variant, _
   Optional ByVal PitchRatio As Variant, _
   Optional ByVal TaperStartRadius As Variant, _
   Optional ByVal TaperEndRadius As Variant _
) As HelixCutout
```

Donde, `HelixAxis` es el eje del cilindro sobre el que se desarrolla la hélice; `NumCrossSections`, es la cantidad de perfiles que se tienen para la elaboración del barrido; `CrossSectionArray`,es el arreglo que contiene el bosquejo a barrer, observe que dicho perfil dependerá del resultado del control de selección; `ProfileSide`,indica el sentido en que se desarrollará el barrido, siendo las opciones posibles las mostradas en la Tabla 5-3; `Pitch`es el paso de la hélice, medido en metros; `NumberOfTurns`,es el número de vueltas desarrolladas por la hélice; `HelixDir`,es el valor que nos permite modificar la dirección en la que se debe ejercer torque para obtener penetración, su valor viene también dada por las dos primeras opciones de la Tabla 5-3.

Corte:La última operación de modelado que se requiere para el proyecto es Corte, con ella se generan los filos de corte del implante.

```
Dim ExtCut As SolidEdgePart.ExtrudedCutout
ExtCut = Me.Model.ExtrudedCutouts.AddFinite(ProfileFilo,_
                                            Lado,_
                                             Lado,_
                                             Dis)
```

Esta operación es sustractiva, lo cual indica que se debe tener una referencia a un modelo ya existente, de lo contrario no se podrá realizar ninguna sustracción. Al igual que el anterior ejemplo, no se crea un segundo objeto, pero este si se encuentra implícito en `Me.Model.ExtrudedCutouts`; tenga en cuenta que `Me.Model` ya se encuentra creado en las operaciones anteriores y en él debe estar almacenado un modelo sobre la cual se realiza el corte. El método para la adición de esta operación es:

```
Public Function AddFinite( _
   ByVal Profile As Profile, _
   ByVal ProfileSide As FeaturePropertyConstants, _
   ByVal ProfilePlaneSide As FeaturePropertyConstants, _
   ByVal Depth As Double _
) As ExtrudedCutout
```

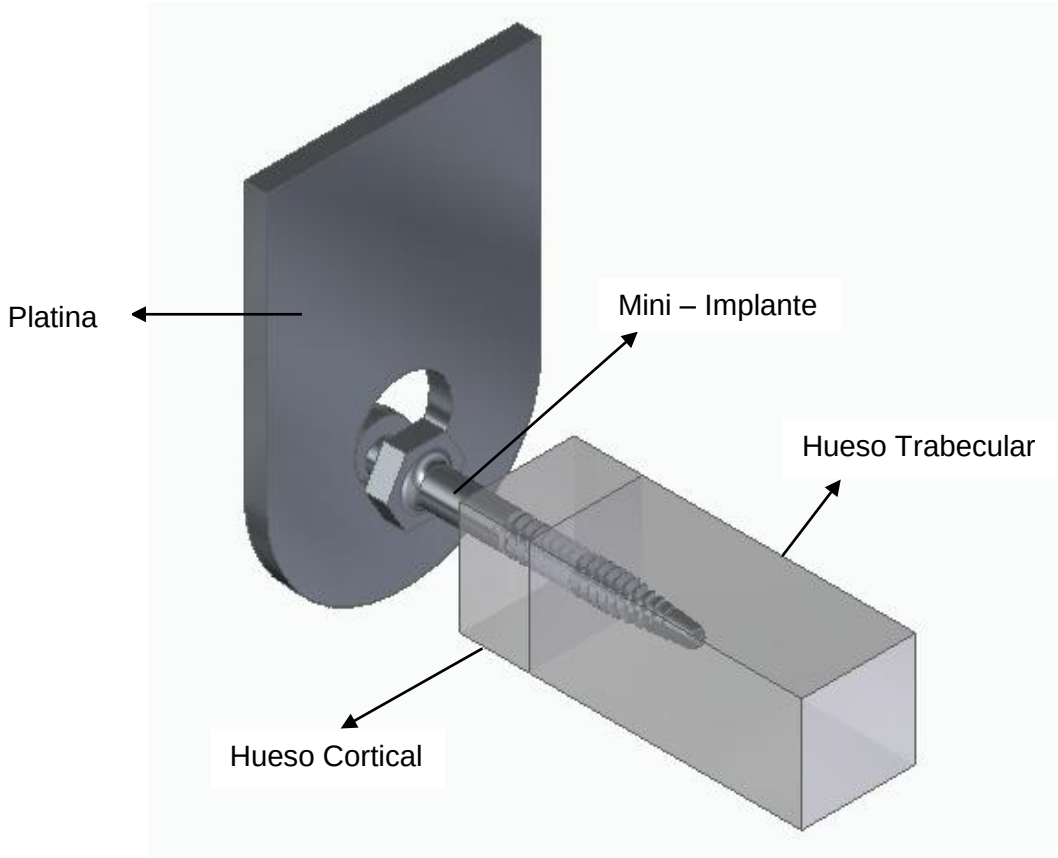
Donde, `Profile`es el perfil del corte, `ProfileSide`indica que si se debe retirar el material al exterior del perfil o al interior de este, `ProfilePlaneSide`,es el sentido en que se debe

realizar el corte; las posibles opciones para estas dos últimas propiedades se encuentran en la Tabla 5-3.

- **Funciones desarrolladas**

Debido a la complejidad en las características biológicas del lugar de anclaje y a las diferentes condiciones de esfuerzos que se pueden presentar en la zona, el producto va dirigido a realizar el análisis estructural por elementos finitos, se requirió crear un conjunto compuesto de implante, hueso cortical, hueso trabecular y platina (VerFigura 5-7) para simular lo más fiel posible a una prueba de tensión. Esto es de importancia ya que por cada grupo de objetos reales, se diseña una clase. Por ende se establecen cuatro clases: Implante, encargada del modelado del mini implante; Hueso encargada del modelado de los huesos cortical y trabecular; Platina encargada del modelado de la platina y Form1 cuya función es el diseño visual y el control de la interfaz de usuario. En el Anexo B se encuentra el diagrama de clases del aplicativo.

Figura 5-7: Conjunto Implante - Hueso - Platina



En la Tabla 5-4se muestra en detalle la función de cada método desarrollado, discretizados por clase.

Tabla 5-4: Métodos desarrollados

Métodos desarrollados por el autor		
Clase	**Método**	**Función**
Implante	CreacionPlano	Crea un plano paralelo a un plano de referencia del *software* CAD en el archivo tipo parte Part. Retorna el plano.
	CreacionPlanoAngulo	Crea un plano en ángulo a una referencia en el archivo parte. Retorna el plano.
	PerfilImplante	Dibuja, sobre un plano recibido como parámetro, las líneas que componen la geometría del implante, con excepción del hexágono.
	CerrarPerfilImplante	Adiciona los *Keypoints*uniendo las líneas que componen el perfil y así, crear la condición de perfil cerrado. Retorna 0 si el perfil cierra correctamente de lo contrario retorna un valor diferente.
	EjeRevolucion	Setea una de las líneas del perfil, especificada en uno de los parámetros, como eje de revolución. Retorna el eje de revolución.
	Revolucion	Este método se encarga de realizar la revolución del perfil del implante. Recibe como parámetro al ángulo de revolución.
	Pf	Calcula el valor de la profundidad del filete, con base al tipo de rosca seleccionada. Retorna el valor de la profundidad.
	PerfilHexagono	Dibuja, sobre un plano recibido como parámetro, las líneas que componen la geometría del hexágono.
	CerrarPerfilHexagono	Adiciona los *Keypoints*uniendo las líneas que

Métodos desarrollados por el autor		
Clase	**Método**	**Función**
		componen el perfil y así, crear la condición de perfil cerrado. Retorna 0 si el perfil cierra correctamente de lo contrario, retorna un valor diferente.
	ExtrusionHex	Encargado de realizar la extrusión del hexágono. Recibe como parámetro la distancia de extrusión.
	PerfilMetrico	Dibuja las líneas del perfil rosca métrica sobre un plano que recibe como parámetro. Además revisa la condición de perfil cerrado.
	PerfilSierra	Dibuja las líneas del perfil rosca en sierra sobre un plano que recibe como parámetro. Además revisa la condición de perfil cerrado.
	Helice	Realiza el barrido – corte – en hélice constante para la rosca hasta la longitud roscada ingresada. Recibe como parámetro el plano del perfil de la rosca y el tipo.
	Helice2	Realiza el barrido – corte – en hélice variable con el ángulo de salida especificado, desde el final de la longitud roscada ingresada hasta la distancia de salida. Recibe como parámetro el plano del perfil de la rosca y el tipo.
	PerfilFilos	Dibuja las líneas del perfil de los filos de corte sobre un plano que recibe como parámetro.
	CerrarPerfilFilos	Adiciona los *Keypoints*uniendo las líneas que componen el perfil y así, crear la condición de perfil cerrado. Retorna 0 si el perfil cierra correctamente de lo contrario retorna un valor diferente.
	ExtrusionFilo	Encargado de realizar la extrusión del hexágono. Recibe como parámetro la distancia

Métodos desarrollados por el autor		
Clase	**Método**	**Función**
		de extrusión.
Hueso - Platina	PerfilCor	Dibuja las líneas del perfil del hueso (cuadrado por simplificación) o platina sobre un plano que recibe como parámetro.
	CreacionPlano	Crea un plano paralelo a un plano de referencia del *software* CAD, en el archivo tipo parte Part. Retorna el plano.
	CerrarPerfil	Adiciona los *Keypoints*uniendo las líneas que componen el perfil y así, crear la condición de perfil cerrado. Retorna 0 si el perfil cierra correctamente de lo contrario, retorna un valor diferente.
	Extrusion	Encargado de realizar la extrusión del hueso o platina. Recibe como parámetro la distancia de extrusión.
	PerfilCut	Dibuja un círculo sobre el hueso (nicho) o platina para un posterior corte, recibe el diámetro del nicho como parámetro.
	ExtrusionCut	Encargado de realizar la extrusión del nicho. Recibe como parámetro la distancia de extrusión.

- **Interfaz gráfica.**

Para una correcta comunicación con el usuario final, se desarrolla una interfaz que sea amigable con el usuario (Ver

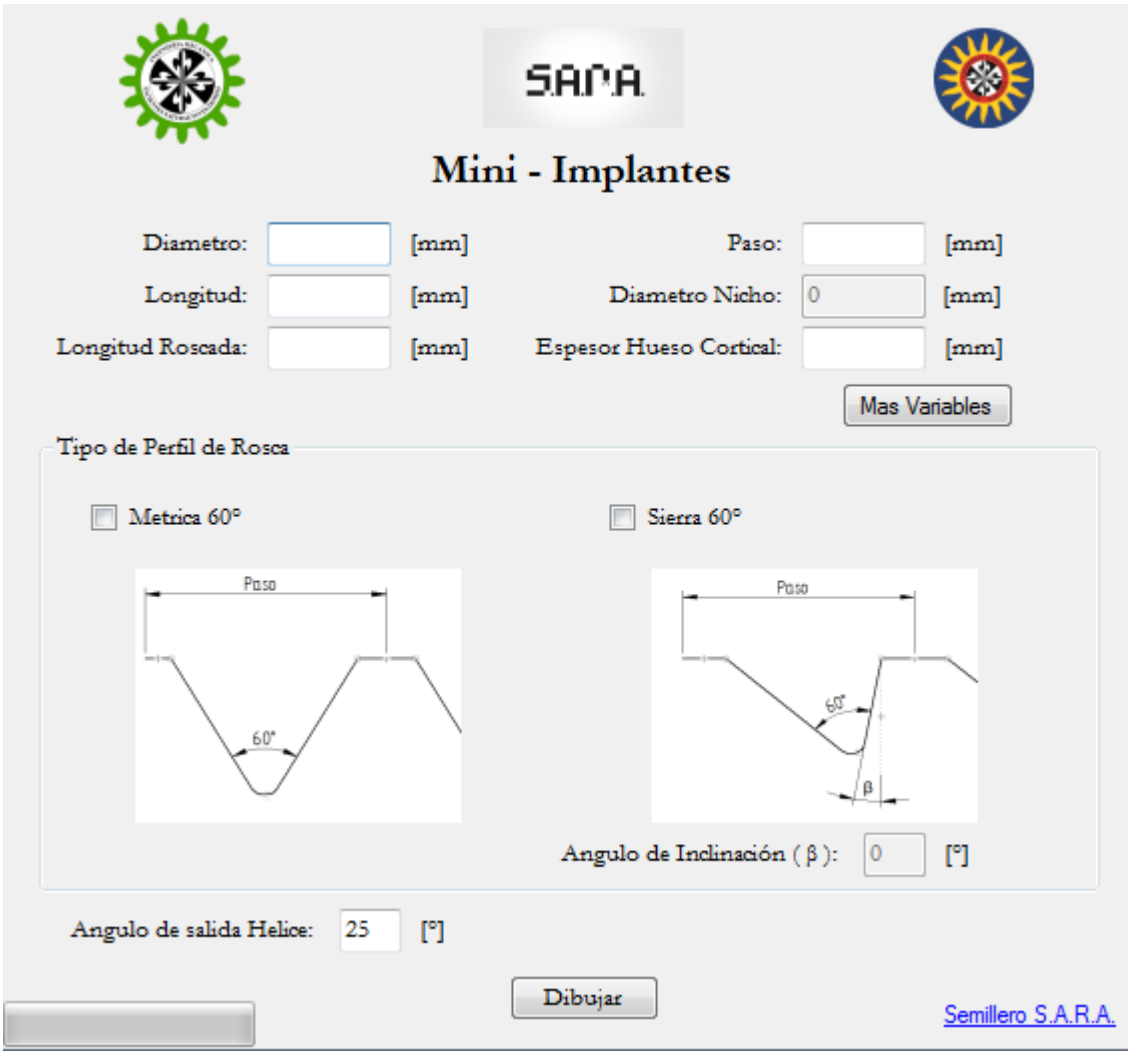
Figura 5-8). Este factor es clave ya que de él dependerá la rápida asimilación de la información, así como una amena estancia durante el tiempo de trabajo con el aplicativo. La programación de esta interfaz se hace mediante el módulode diseñode Visual Basic que contiene todo tipo de herramientas como controles, comandos, textos, interacciones,

ventanas, etc., donde un solo clic sostenido sobre estos y dirigiéndolos al telar del formulario, es suficiente para agregarlos con las configuraciones básicas.

A través de esta interfaz, se recolectan los parámetros que junto con el proyecto de diseño, se establecen como necesarios para parametrizar. Ademas, se establecieron restricciones a los campos de escritura, con la función de evitar sobredimensionamientos y errores en el programa. En términos de diseño se restringieron: Diametro (1 – 2 mm), Longitud (5 – 12 mm), Longitud Roscada (< Longitud), Paso (0.1 – 1 mm), Espesor Hueso Cortical (0.1 – 5 mm), Angulo de Inclinación (5° - 30°); en términos de manufactura: Angulo de salida Helice (5° - 30°); en términos de programación, se restringio, en todos los campos, el uso de la coma como punto decimal y la selección simultanea de los dos tipos de rosca.

Se utilizan los controles Labels, TextField, CheckBox, Buttons, Image; a todos ellos se les crean métodos de inspección de datos con el objetivo de prevenir errores, causados por mala digitación. En el Anexo A se encuentra la clase elaborada para la interfaz gráfica.

Figura 5-8: **Interfaz gráfica API**



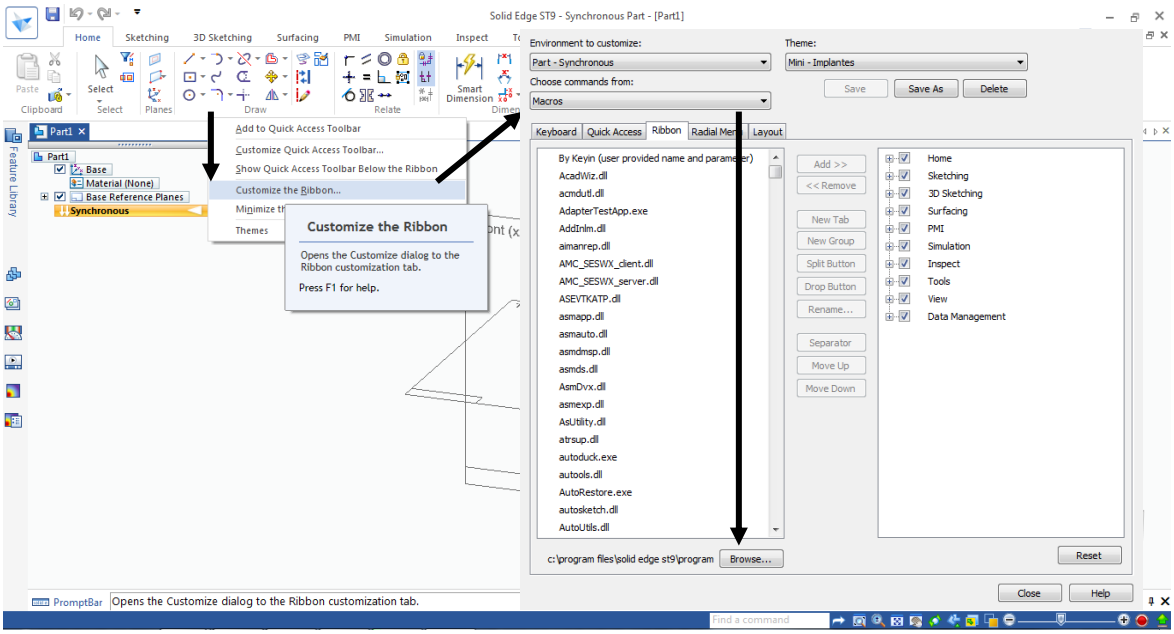
5.2.3 Comunicación API *Software* CAD

Terminado el aplicativo, este debe ser adecuado en el entorno de Solid Edge a través de un botón, ubicado en la barra de comandos. La adición de botones, es realizada por medio de la ventana de personalización. Es creado un grupo de botones, con el nombre “S.A.R.A.”, dentro de la sección Home de un archivo tipo parte y en este se adiciona la Api, renombrándola como Mini – Implantes.

El procedimiento a seguir es (verFigura 5-9):1. Dar clic en cualquier botón de la barra de comandos de un archivo ISO *Metric Part*, esto abrirá una lista de opciones; 2. Seleccionar la opción *Customize the Ribbon*, que abre la ventana del personalizador; 3. Elegir Macros en el campo *Choose commands from* (recuadro Figura 5-9); 4. Dar clic en

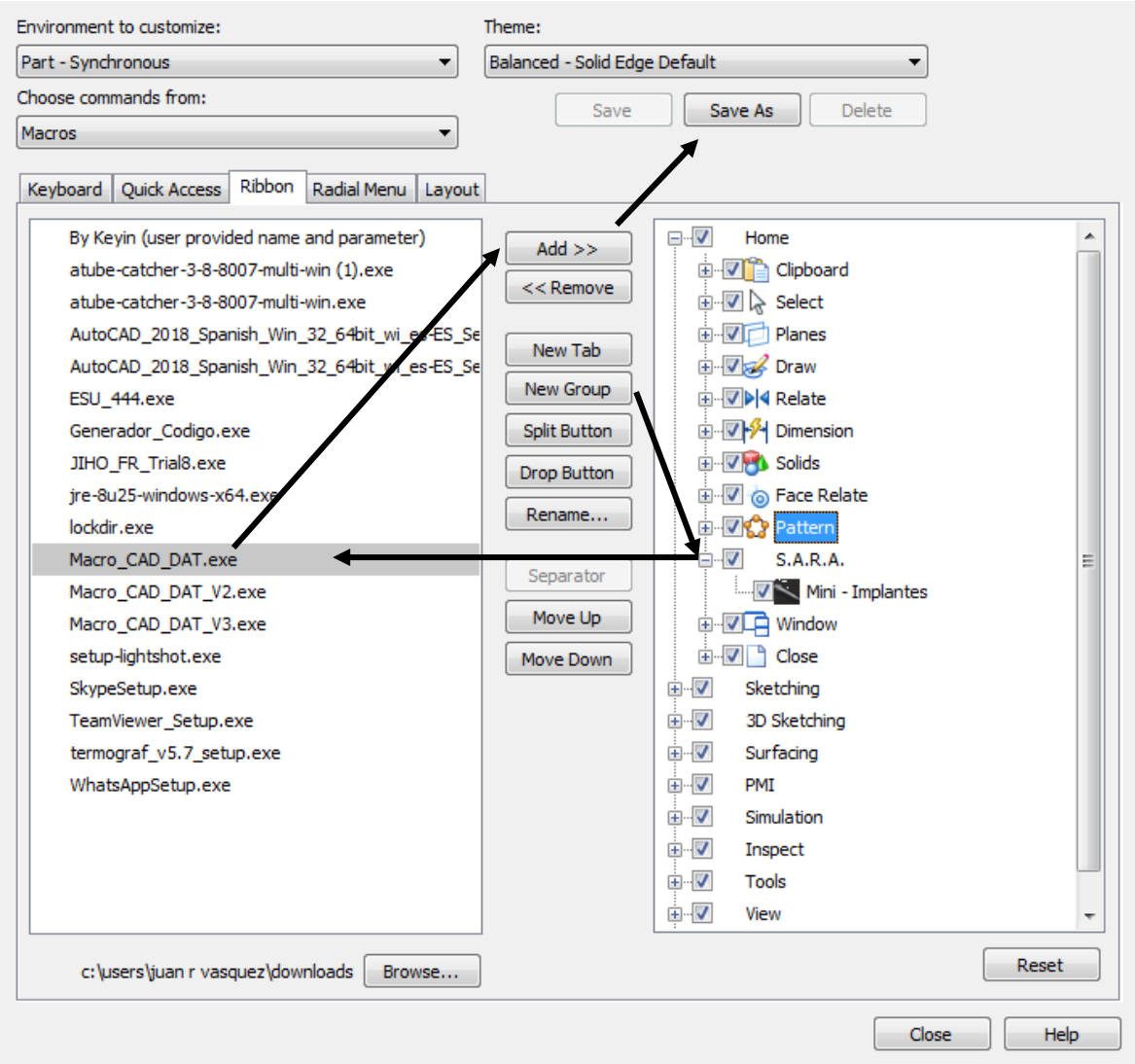
el botón Browser y buscar la carpeta que contiene el ejecutable del Api, luego dar clic en aceptar.

Figura 5-9: Creación botón y enlace con API – 1.



Con lo anterior, quedan cargados todos archivos tipo .exe y .dll en la sección izquierda del personalizador (verFigura 5-10). Posterior a esto, se sigue el siguiente procedimiento: 1. Crear un nuevo grupo de botones dando clic en *New Group* y teniendo seleccionado el grupo *Pattern*, esto con el fin de que el nuevo grupo quede ubicado después de este; 2. Renombrar con “S.A.R.A.” el nuevo grupo por medio de la opción *Rename* (clic derecho); 3. Seleccionar el ejecutable del Api, ubicado en la sección izquierda del personalizador, y dar clic en *Add>>*, ubicándolo en el grupo “S.A.R.A.”; 4. Renombrar con “Mini – Implantes”y con clic derecho dar la propiedad de *Large Button*; 5. Guardar el nuevo tema; 6. Cerrar.

Figura 5-10: Creación botón y enlace con API– 2.



Al finalizar los pasos descritos anteriormente, el resultado debe ser algo como lo mostrado en laFigura 5-11

Figura 5-11: Api adecuada al entorno de Solid Edge

A partir de allí, cada vez que se dé clic sobre el botón Mini – Implantes aparecerá la interfaz gráfica de la

Figura 5-8y se podrán dibujar cualquier mini implante del conjunto de configuraciones posibles.

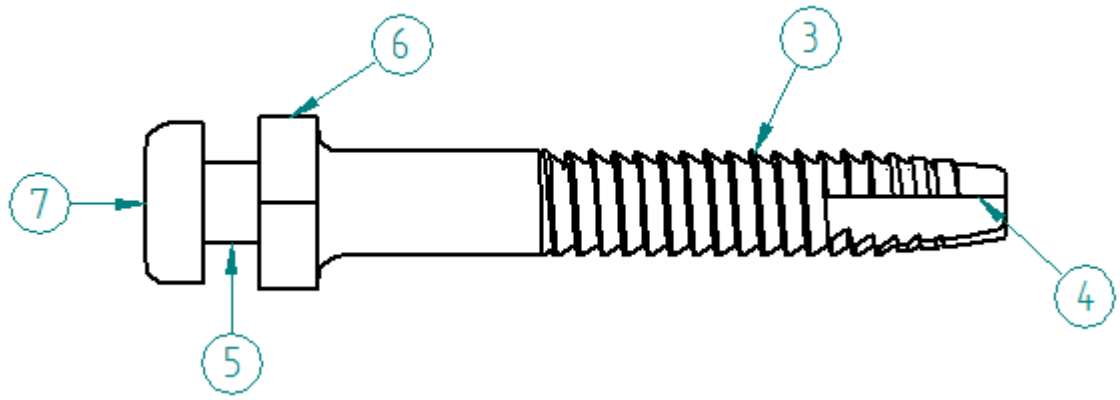
6. Plan de Manufactura

En términos de manufactura, la fabricación de mini implantes dentales, denota un desafío para lograr una producción con altos niveles de flexibilidad y altos volúmenes de fabricación. En este capítulo se describe el proceso implementado para la manufactura de mini implantes, en la primera parte se analiza el material y geometría de la pieza, concluyendo el tipo de maquina a utilizar junto con las operaciones tecnológicas requeridas para su manufactura. Se muestra la hoja de proceso, donde se indican las condiciones de corte y el orden operacional. Por último se aborda solución implementada para la obtención de un proceso de manufactura flexible pero continuo.

6.1 Análisis de geometría, Operaciones tecnológicas y elección de maquinaria

Todo proceso de manufactura inicia con el análisis de la geometría y material de la pieza a elaborar, con esta información se define qué tipo de proceso, maquinaria y herramental es el indicado. Los mini implantes dentales son fabricados en aleación de titanio (Ti-6Al-4V Grado 23) teniendo mejores resultados en la oseointegración y mejor aceptación por el cuerpo humano. En laFigura 6-1se observa los planos de manufactura del mini implante dental a fabricar, con las características adaptables a cada paciente, determinadas después de un proceso de diseño por elementos finitos.

Figura 6-1: Geometría mini implante dental



Operaciones Tecnológicas	
1	Desbaste en cilindrado
2	Acabado en cilindrado
3	Roscado
4	Fresado filos de corte
5	Ranurado diametral
6	Fresado hexágono
7	Tronzado

Como se observa en la Figura 6-1, por su geometría en revolución y roscado, requiere de una máquina - herramienta tipo torno,adicionalmente, se necesitanoperaciones de fresado para la obtención de los filos de corte y el hexágono. EnlaFigura 6-1se detallan las operaciones tecnológicas a realizar para su manufactura; por el número y clase de operaciones, la precisión y las dimensiones particularmente pequeñas,se hace necesaria la utilización de una máquina especializada.

El torno tipo suizo multieje Polygim 20CSB, cumple a cabalidad con dichos requisitos ya que por su arquitectura, es posible ejecutar la totalidad de las operaciones tecnológicas del implante. La máquina herramienta cuenta con 5 grados de libertad y 7 ejes de movimiento, 6 herramientas de torneado externo y 20 herramientas de fresado con ataque frontal o cruzado, siendo algunas de estas motorizadas con hasta 6000 rpm; cuenta con un controlador Mitsubishi con capacidades de programación paramétrica.

Este proyecto implicó la compra de herramental,para operaciones específicas de maquinado. Tal fue el caso del inserto para ranurado de 0.5 mm de espesor, permitiendo la realización de la ranura (Ver Figura 6-1), junto con su respectivo porta inserto. Compra de la pinza de sujecióny buje guía (VerFigura 6-2) para materia prima (titanio ¼"), ya que solo se disponía del sistema de sujeción para ¾". Además, se adquirió un inserto para tronzado de 1.5 mm de espesor, con el objetivo de reducir el desperdicio de material en el proceso de tronzado de un 30% hasta un 15%.Para las operaciones tecnológicas de desbaste y acabo en cilindrado, de roscado, de filos de corte y planeados del hexágono ya se disponía de las herramientas requeridas. En laTabla 6-1 se encuentra desglosado por operación, cada una de las herramientas utilizadas.

Figura 6-2: Herramientas adquiridas. Izq, pinzas de sujeción de materia prima de ¼". Der, inserto de 1.5 para tronzado ref

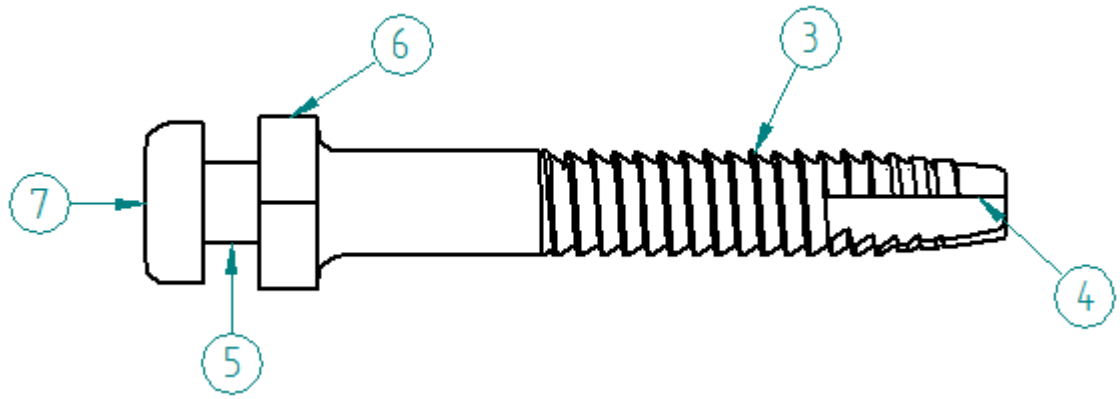
Tabla 6-1: Herramental utilizado en el mecanizado del mini implante

Operación	Herramienta	P.M	Características	Referencia
Desbaste	Inserto	T0101	Radio: 0.8, Kyocera	DCMT11T308GK
Acabado	Inserto	T0404	Radio: 0.4, Sandvik	VBMT110302PF
Roscado	Inserto rosca	T0505	Metrica a 60° Radio: 0.05	TTX32R60005
Planos filos de corte	Fresa	T0808	Diametro: 6 mm 4 filos	
Ranurado	Inserto	T0606	Ancho: 0.5 mm	NG2M050RK KC5025
Planos hexágono	Fresa	T0808	Diametro: 6 mm 4 filos	
Tronzado	Inserto	T0303	Ancho: 1.5 mm	NGD2M150RKKC5025

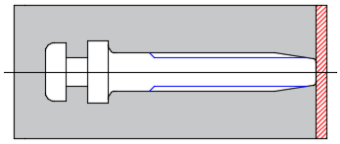
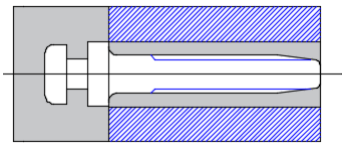
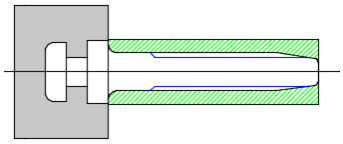
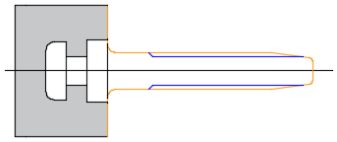
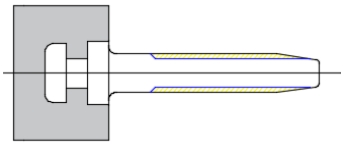
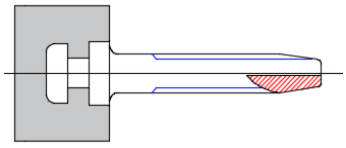
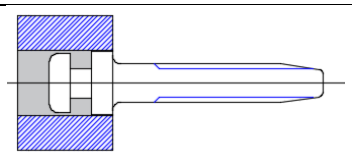
6.2 Hoja de proceso

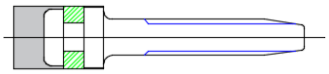
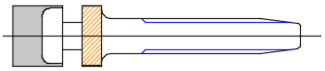
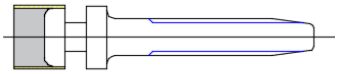
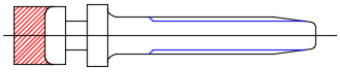
Seleccionadas la máquina y herramientas para el mecanizado, se procede a la elaboración de la hoja de procesos, describiendo los pasos para la fabricación del lote de mini implantes. A continuación se muestra la hoja de proceso.

<table>
<tr><td colspan="3">UNIVERSIDAD SANTO TOMAS
DIVISIÓN DE INGENIERÍAS
FACULTAD DE INGENIERÍA MECÁNICA
HOJA DE PROCESO DE MECANIZADO</td></tr>
<tr><td colspan="2"></td><td>PÁGINA:
60 de 122</td></tr>
<tr><td>PIEZA:
Mini implante dental</td><td>MATERIAL:
Titanio Ti – 6Al – 4V</td><td>DIMENSIONES EN BRUTO:
¼”x 1 m</td></tr>
<tr><td colspan="3">PREPARACIÓN PREVIA DEL MATERIAL:
Ninguna</td></tr>
<tr><td colspan="2">MÁQUINA:
Torno Suizo Polygim Diamond 20 CSB</td><td rowspan="2">HERRAMIENTAS, DISPOSITIVOS DE FIJACIÓN E INSTRUMENTOS:

Tabla 6-1: Herramental utilizado en el mecanizado del mini implante</td></tr>
<tr><td colspan="2">INSUMOS:</td></tr>
<tr><td colspan="2">ELABORÓ: Juan David Ramírez Vásquez</td><td>REVISÓ: Ing. García Barbosa Jorge Andrés</td></tr>
<tr><td colspan="3">PLANO:
 
</td></tr>
</table>

Nota: El número designa la operación
NM: Revoluciones máximas

#	Operación	Croquis	Útil	Cond. de corte	Tiempo [s]
1	Refrentado		T0101	V: 40 m/min NM: 4000 rpm Ap: 0.5 mm f: 0.05 mm/rev	6.24
2	Desbaste 1A		T0101	V: 40 m/min NM: 4000 rpm Ap: 0.6 mm f: 0.05 mm/rev	29.78
3	Desbaste 1B		T0404	V: 40 m/min NM: 4000 rpm Ap: 0.2 mm f: 0.05 mm/rev	31.38
4	Acabado		T0404	V: 50 m/min NM: 4000 rpm Ap: 0.2 mm f: 0.03 mm/rev	11.71
5	Roscado		T0505	V: 4.7 - 6.3 m/min N: 1000 rpm Ap: 0.05 mm f:0.2,0.8 mm/rev	12.19
6	Fresado filos de corte		T0808	V: 28.27 m/min N: 1500 rpm Ap: 1.5 – 2 mm f: 50 mm/min	47.15
7	Desbaste 2		T0101	V: 40 m/min NM: 4000 rpm Ap: 0.6 mm	29.23

				f: 0.05 mm/rev	
8	Ranurado		T0606	V: 30 m/min NM: 4000 rpm Ap: 0.1 mm f: 0.02 mm/rev	17.93
9	Fresado hexágono		T0808	V: 28.27 m/min N: 1500 rpm Ap: 0.1 mm f: 50 mm/min	90.80
10	Desbaste 3		T0404	V: 40 m/min NM: 4000 rpm Ap: 0.2 mm f: 0.03 mm/rev	12.33
11	Tronzado		T0303	V: 30 m/min NM: 2500 rpm Ap: 0.3 mm f: 0.01 - 0.03 mm/rev	21.43

6.3 Código G, programación paramétrica

El torno tipo suizo Polygim Diamond 20 CSB posee un controlador Mitsubishi, el cual se programa en estándar ISO(verTabla 6-2). Se programa en código G, conforme a la geometría de la pieza, una de las posibles configuraciones del mini implante con características geométricas específicas. Debido a que las modificaciones en los requerimientos funcionales son en valor y no en forma y que la secuencia de su proceso de fabricación no varía, las múltiples configuraciones de mini implantes conforman una familia de partes. Así, parte de la problemática puede ser abordada a través de la programación paramétrica del código, identificando la cantidad de variables que se van a utilizar y en qué partes se ubicarán. En seguida, se encuentra el código realizado.

Tabla 6-2: Códigos G/M utilizados

Códigos Geometricos		Códigos Miscelaneos	
G0	Movimiento Rapido	F	Avance
G1	Interpolación lineal	M3	Enciende husillo CW
G2	Interpolación circular CW	M5	Apaga husillo
G3	Interpolación circular CCW	M8	Refrigerante automático, encendido
G18	Selección plano Z – X	M9	Refrigerante automático, apagado
G19	Selección plano Y – Z	M26	Activa bandeja de recolección
G21	Programación en mm	M27	Retrae bandeja de recolección
G28	Retorno punto de referencia	M30	Fin de programa
G40	Comp. radio herramienta, cancelada	M90	Desenganchar eje C
G41	Comp. Radio herramienta, izquierda	M92	Enganchar eje C
G43.1		M98	Llamado a subprograma
G50	Seteo de ejes, limitante de revoluciones	M99	Retorno a programa principal
G54	Cero de pieza	M103	Enciende herramientas motorizadas CW
G70	Ciclo de acabado	M105	Apaga herramientas motorizadas
G71	Ciclo de torneado longitudinal	S1	Velocidad husillo, velocidad de corte
G75	Ciclo de ranurado longitudinal	T	Herramienta
G76	Ciclo de roscado		
G96	Velocidad de corte constante, activada		
G97	Velocidad de corte constante, cancelada		
G98	Avance en mm/rev		
G99	Avance en mm/rev		

```
% (UNIVERSIDAD SANTO TOMAS)
```

```
N10(REFRENTADO)
G54G21G99G50S1=4000
```

```
G43.1
G28U0
G28V0
T0101
G0Y0
M8
G96M3S1=40
G0X8Z0
G1X-1Z0F0.05
G1Z-1
G0X20

N20(DESBASTE1A)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0101
G0Y0
G96M3S1=40
G0X8Z-2
G71U0.6R0.5F0.05
G71P21Q22U0W0
N21G0X3.08
G1Z -7
N22G1X8

N30(DESBASTE1B)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0404
G0Y0
G96M3S1=40
G41
G0X4Z-2
G71U0.2R0.5F0.05
G71P31Q32U0.2W0
N31G0X0
G1Z0
G1X0.945,R0.2
G1X 1.5 Z2.258
G1Z -7 ,R0.4
N32G1X4
G40

N40(ACABADO)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0404
G0Y0
G96M3S1=50
G0X7Z-1
G41
G70P31Q32F0.03
G40

N50(ROSCADO)
G21G99G50S1=2000
G43.1
G28U0
G28V0
T0505
G0Y0
G97M3S1=1000
G0X2.5Z-1
G76P020560Q0.02R0.1
G76X[ 1.5 -2*0.177]Z 5.5P0.177Q0.08F
0.3

N60(FILOS)
G21G98G19G50S3=2000
G43.1
M92
G28U0
G28V0
M5
M90
T0808
M8
G0Z-4
Y0
G97M103S3=1500

G0C0
Z-4Y[3+[1.5 /2]-0.5]
X0.1
G1Z1.5F50
G1Z-4

G0C180
Z-4Y[3+[ 1.5 /2]-0.5]
X0.1
G1Z1.5F50
G1Z-4
M105
M92
G0X100

N70(DESBASTE2)
G18G99G50S1=4000
G43.1
G28U0
G28V0
T0101
```

```
G0Y0
G96M3S1=40
G0X8Z[  7  - 2]
G71U0.6R0.5F0.05
G71P61Q62U0W0
N71G1X3.08
G1Z[  7    + 4.6]
N72G1X8

N80(RANURADO)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0606
G0Y0
G96M3S1=30
G0X4Z[  7 + 1.5]
G75R0.1F0.02
G75X2P0.1
G0X4Z[  7 + 1.75]
G75R0.1F0.02
G75X2P0.1
G0X4Z[  7 + 2]
G75R0.1F0.02
G75X2P0.1

N90(HEXAGONO)
G21G98G19G50S3=2000
G43.1
M92
G28U0
G28V0
M5
M90
T0808
M8
G0Y4.5
G97M103S3=1500
G0Z[ 7  - 3.5]
X-4

G1Z[  7 + 1.2]F50
G1Z[  7  - 3.5]

G0C60
G1Z[  7  + 1.2] F50
G1Z[  7  - 3.5]

G0C120
G1Z[  7  + 1.2] F50
G1Z[  7  - 3.5]

G0C180
G1Z[  7  + 1.2] F50
G1Z[  7  - 3.5]

G0C240
G1Z[  7  + 1.2] F50
G1Z[  7  - 3.5]

G0C300
G1Z[  7  + 1.2] F50
G1Z[  7  - 3.5]

M105
M92
G0X100

N100(DESBASTE3)
G21G18G99G50S1=4000
G43.1
G28U0
G28V0
T0404
G0Y0
G96M3S1=40
G0Z[  7 + 1.8]
G0X7
G1X2.9F0.03
G1Z[  7    + 4.6]
G1X7

N110(TRONZADO)
G21G99G50S1=2500
G43.1
G28U0
G28V0
T0303
G0Y0
G96M3S1=30
G0X7Z[  7    + 4.5]
G75R0.1F0.03
G75X1.6P0.3
G1X4
G1Z[  7    + 4]
G1X2.9
G2X1.9Z[  7    + 4.5]R0.8
M26
G4X1
G75R0.1F0.01
G75X-1P0.3
M27
G0X50
G0Z[  7    + 4.7]
G50Z0
M9
M30
```

%

Se identifica qué parámetros de la geometría se quieren y pueden realizar, desde el punto de vista de manufactura y recursos. Las características a parametrizar son: Diámetro, Longitud, Longitud roscada y paso. El tipo de rosca no fue parametrizable, debido a que comercialmente, no fue posible adquirir insertos con diferentes perfiles de rosca para pasos inferiores a 1mm; pasos habituales en los implantes dentales.

En el código anterior, se identifican con recuadros de colores los lugares donde es posible la utilización de variables. Cada color representa un parámetro de la siguiente manera:

: Diámetro

: Longitud

: Longitud roscada

: Paso

Además, se pueden utilizar variables auxiliares que servirán para hacer cálculos matemáticos, estas son:

: Diámetro filos de corte

: Profundidad del filete

: Longitud final cono

El controlador Mitsubishi *Meldas 60S series*, tiene la capacidad de ejecutar códigos programados paramétricamente, para ello, maneja variables que permiten aumentar la flexibilidad, comúnmente utilizadas en familia de partes; se identifican por el símbolo numeral (#) seguido de un número. Posee dos tipos de variables, comunes y locales, entre las diferencias de estas,se pueden encontrar, que las variables comunes mantienen su valor así el controlador sea apagado, caso contrario de las locales, en cuyo caso son llevadas a cero. Otra diferencia es su identificación, siendo las variables comunes identificadas con la serie #500's y las locales con la serie #100's

Teniendo en cuenta lo anterior, se utilizan variables comunes para los parámetros y variables locales para parámetros auxiliares, entonces:

#500= Diámetro

#501= Longitud

#502= Longitud roscada
#503= Paso
#100= Diámetro filos de corte
#101= Profundidad del filete
#102= Longitud final cono

6.4 Código G, Programación con subprogramas

Como se mencionó, parte del problema es solucionado con estas pequeñas alteraciones al código G, obteniendo un proceso capaz de producir diferentes configuraciones de mini implantes, pero sin continuidad, yaque finalizada la fabricación de una pieza, se debe parar la máquina y modificar manualmente las variables comunes en el controlador, y así poder proseguir con la fabricación de la nueva pieza.

Para la obtención de un proceso de manufactura flexible y continuo, se utilizó la técnica de programación por subprogramas. Esta consiste en un código principal, desde el cual se hace el llamado a subprogramas con rutinas específicas; es posible invocar diferentes subprogramas y cuantas veces sea necesario. El controlador Mitsubishi realiza los llamados por medio de los códigos misceláneos M98 y M99; el primero es acompañado del parámetro P, que invocará al subprograma identificado con el número indicado (ej.: M98 P20). El segundo se utiliza para regresar al código principal.

Siguiendo esta técnica, el código G mostrado anteriormente, hace las veces de subprograma y tendrá contenida la rutina parametrizada, para la fabricación de las diferentes configuraciones de mini implantes. Para obtener la propiedad de subprograma, primero, deben ser realizadas pequeñas modificaciones; 1. Sustituir el comando M30 por M99, 2. Adicionar en sus primeras líneas las expresiones matemáticas para el cálculo de las variables auxiliares. Este código estará almacenado en la memoria del controlador, no requiere ser modificado durante la ejecución del proceso de mecanizado y se denominara código residente, a continuación se muestra la versión final.

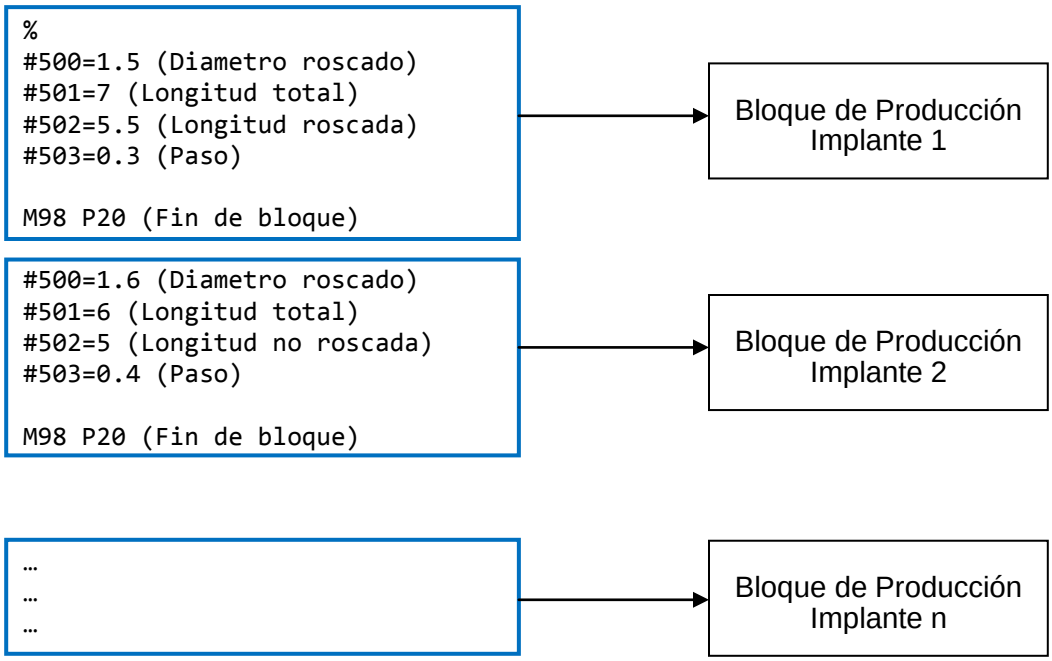
```
%                                    (UNIVERSIDAD SANTO TOMAS)
```

```
(VARIABLESAUXILIARES)
#101=[[[7*SQRT[3]]/16]*#503]-0.05
#100=#500-[2*#101]-0.2
#102=[[#500-#100]/2]*TAN[83]

N10(REFRENTADO)
G54G21G99G50S1=4000
G43.1
G28U0
G28V0
T0101
G0Y0
M8
G96M3S1=40
G0X8Z0
G1X-1Z0F0.05
G1Z-1
G0X20

N20(DESBASTE1A)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0101
G0Y0
G96M3S1=40
G0X8Z-2
G71U0.6R0.5F0.05
G71P21Q22U0W0
N21G0X3.08
G1Z#501
N22G1X8

N30(DESBASTE1B)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0404
G0Y0
G96M3S1=40
G41
G0X4Z-2
G71U0.2R0.5F0.05
G71P31Q32U0.2W0
N31G0X0
G1Z0
G1X#100,R0.2
G1X#500Z#102
G1Z#501,R0.4
N32G1X4
G40

N40(ACABADO)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0404
G0Y0
G96M3S1=50
G0X7Z-1
G41
G70P31Q32F0.03
G40

N50(ROSCADO)
G21G99G50S1=2000
G43.1
G28U0
G28V0
T0505
G0Y0
G97M3S1=1000
G0X2.5Z-1
G76P020560Q0.02R0.1
G76X[#500-
2*#101]Z#502P#101Q0.08F#503

N60(FILOS)
G21G98G19G50S3=2000
G43.1
M92
G28U0
G28V0
M5
M90
T0808
M8
G0Z-4
Y0
G97M103S3=1500

G0C0
Z-4Y[3+[#500/2]-0.5]
X0.1
G1Z1.5F50
G1Z-4

G0C180
Z-4Y[3+[#500/2]-0.5]
X0.1
G1Z1.5F50
G1Z-4
M105
M92
G0X100
```

```
N70(DESBASTE2)
G18G99G50S1=4000
G43.1
G28U0
G28V0
T0101
G0Y0
G96M3S1=40
G0X8Z[#501-2]
G71U0.6R0.5F0.05
G71P61Q62U0W0
N71G1X3.08
G1Z[#501+4.6]
N72G1X8

N80(RANURADO)
G21G99G50S1=4000
G43.1
G28U0
G28V0
T0606
G0Y0
G96M3S1=30
G0X4Z[#501+1.5]
G75R0.1F0.02
G75X2P0.1
G0X4Z[#501+1.75]
G75R0.1F0.02
G75X2P0.1
G0X4Z[#501+2]
G75R0.1F0.02
G75X2P0.1

N90(HEXAGONO)
G21G98G19G50S3=2000
G43.1
M92
G28U0
G28V0
M5
M90
T0808
M8
G0Y4.5
G97M103S3=1500
G0Z[#501-3.5]
X-4

G1Z[#501+1.2]F50
G1Z[#501-3.5]

G0C60
G1Z[#501+1.2]F50
G1Z[#501-3.5]

G0C120
G1Z[#501+1.2]F50
G1Z[#501-3.5]

G0C180
G1Z[#501+1.2]F50
G1Z[#501-3.5]

G0C240
G1Z[#501+1.2]F50
G1Z[#501-3.5]

G0C300
G1Z[#501+1.2]F50
G1Z[#501-3.5]

M105
M92
G0X100

N100(DESBASTE3)
G21G18G99G50S1=4000
G43.1
G28U0
G28V0
T0404
G0Y0
G96M3S1=40
G0Z[#501+1.8]
G0X7
G1X2.9F0.03
G1Z[#501+4.6]
G1X7

N110(TRONZADO)
G21G99G50S1=2500
G43.1
G28U0
G28V0
T0303
G0Y0
G96M3S1=30
G0X7Z[#501+4.5]
G75R0.1F0.03
G75X1.6P0.3
G1X4
G1Z[#501+4]
G1X2.9
G2X1.9Z[#501+4.5]R0.8
M26
G4X1
G75R0.1F0.01
```

```
G75X-1P0.3
M27
G0X50
G0Z[#501+4.7]
G50Z0
M9
M99
%
```

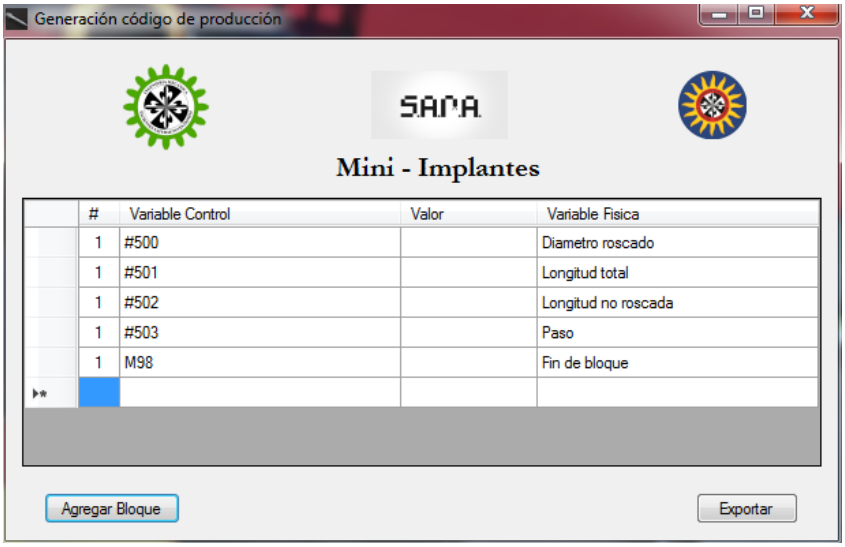
Con el subprograma listo, solo hace falta el código de producción, desde el cual se hará el llamado al residente. Tiene la función de modificar el valor de las variables comunes (Características geométricas mini implante) y la de llamar al código residente, una vez finalizada la producción de la pieza, retornará para modificar de nuevo las variables y hacer el siguiente llamado; será variante y su tamaño dependerá del lote de piezas a producir. Su estructura es la siguiente:




```
M30
%
```

Como se observa, cada bloque corresponde a una configuración de un mini implante y se escribirán un numero de bloques, iguales a la cantidad de mini implantes a producir, siendo esto una gran problemática, para producciones grandes, debido a la cantidad de líneas que se requieren digitar. Por ello, se desarrolla un aplicativo en Visual Studio, con la capacidad para generar un txtque posea la estructura de un código de producción. En la Figura 6-3se observa la interfaz del aplicativo computacional.

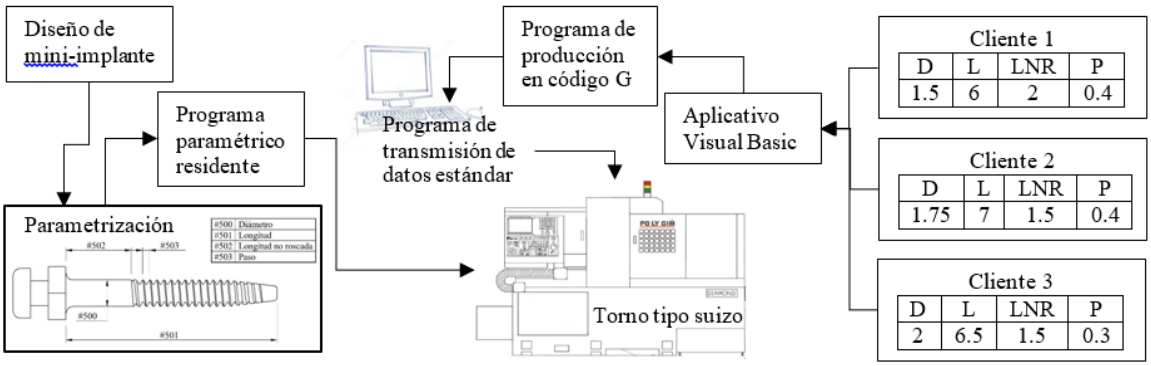
Figura 6-3: Aplicativo generación código de producción.



En el aplicativo se hace la recopilación de los requerimientos de cada cliente a través de un control de muestreo de datos en formato tabla (*Data View Grid*). Obtenido el programa de producción para la fabricación de un lote de piezas, el aplicativo genera un archivo en código G, que es enviado a través del puerto de comunicación RS232 al controlador de la máquina, por medio de un programa de transmisión de datos estándar. El código del programa de producción es sustituido por uno nuevo, una vez finalizado el proceso de manufactura. Con la programación por subprogramas se elimina el cambio de variables en el controlador de forma manual, dando continuidad al proceso. Las modificaciones realizadas reducen el tiempo de producción de 7.84 min a 5.17 min.

La Figura 6-4 presenta el resumen de la metodología de implementación de un proceso de manufactura flexible y continuo para la obtención de lotes de familias de partes con diferentes características geométricas (Ver Anexo D).

Figura 6-4: Metodología de implementación de un proceso de manufactura flexible y continuo. D: Diámetro, L:Longitud, LNR: Longitud no roscada, P: Paso.



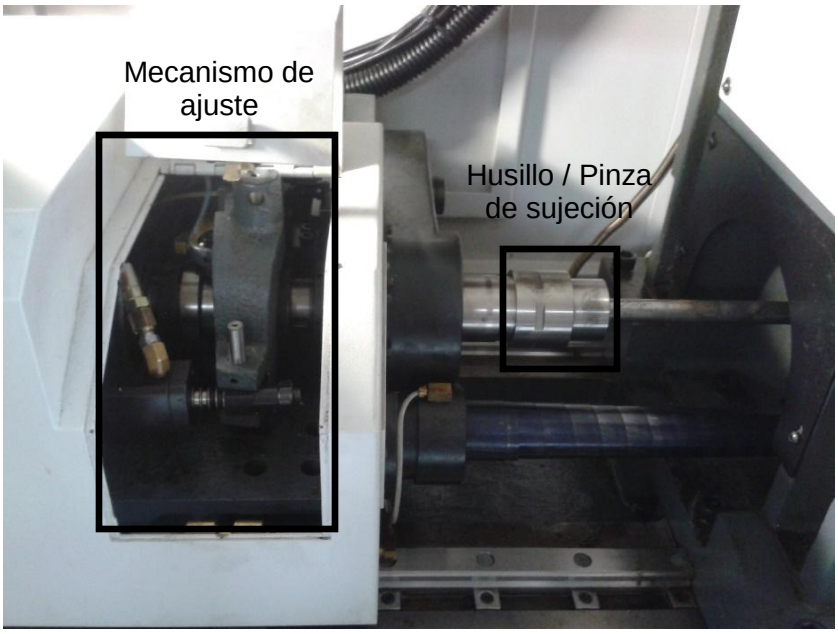
7. Montaje, fabricación y metrología

El buen diseño y planificación de un proceso de manufactura, tiene como resultado el buen desarrollo del mismo en la etapa de manufactura. En este capítulo encontrará los procedimientos de montaje de pinzas y bujes de sujeción, al igual que del herramental requerido. También, se incluye el código de producción obtenido del aplicativo desarrollado en el capítulo 2 y la manera como fue ingresado al controlador de la máquina. Por último, está el procedimiento de metrología y compensación de herramientas, desarrollado con el objetivo de obtener un producto de calidad.

7.1 Montaje y setup de la maquina

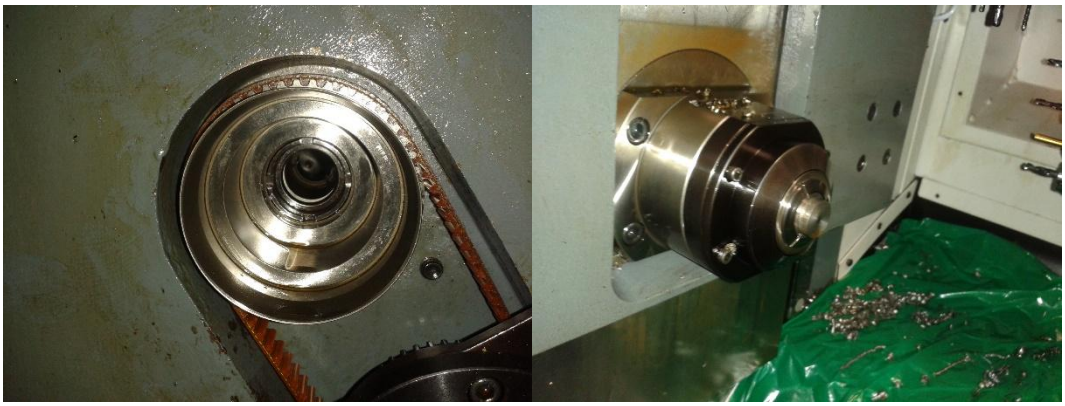
El primer acondicionamiento que se realiza al torno Poligym es el montaje de la pinza y el buje de sujeción de materia prima (Ver Figura 6-2). Los elementos adquiridos son para barras cilíndricas de ¼" de diámetro y están ubicados como se indica en laFigura 7-1.De primero, es instalada la pinza de sujeción, esto se realiza con la ayuda de dos llaves especiales; la primera ejerceuna traba al movimiento rotatorio y libre que posee el husillo, y la segunda es utilizada para realizar palanca yasí soltar la tuerca. Al interior de este se encuentra la pinza que se retira manualmente, se realiza una limpieza, se cambia la pinza y se ajusta de nuevo la tuerca del husillo. Se debe comprobar que la presión de agarre de la pieza sea la adecuada, en caso de no serlo, se debe ajustar con el mecanismo mostrado en la Figura 7-1.

Figura 7-1: Instalación pinza de sujeción y Mecanismo de ajuste presión pinza



En el caso del buje de sujeción, se debe retirar por la parte posterior del torno. Con la ayuda de la herramienta especializada se afloja la tuerca que sujeta el buje (Ver Figura 7-2), se limpia el área, se instala el nuevo buje y se ajusta la tuerca; la fuerza de apriete será la necesaria para que el buje quede en su posición, pero no demasiada que evite el libre paso de la materia prima.

Figura 7-2: Buje de sujeción. Izq vista posterior, Der vista anterior



El montaje de las herramientas se realiza en el orden indicado en la hoja de procesos y en la Tabla 6-1. Para la instalación de las herramientas de torneado, se debe retirar los pernosde la cuña respectiva a la posición de la herramienta en el magazín. Debido a que

estas cuñas actúan a presión, se recomienda colocar uno de los pernos retirados, en alguno de los agujeros auxiliares de las cuñas, para de tal manera ejercer presión contraria y así desprender la cuña. Es de tener cuidado no dejar caer ningún elemento al fondo del torno, ya que su recuperación puede ser tediosa, además de poder provocar un daño en el herramental.

Luego de una limpieza al área, se debe llamar a la herramienta desde el controlador y posicionarla en el diámetro del material. Con esto se podrá descansar la herramienta sobre la materia prima y no se alterara las compensaciones ya existentes. Ubicado el magazín, se procede a la instalación de la nueva herramienta, colocándola junto con la cuña y posteriormente ajustando los pernos.Por último, se debe retirar la herramienta en sentido positivo en el eje X, de lo contrario, el inserto podría sufrir daños al estrellarse con el material. Este procedimiento se realiza con todas las herramientas requeridas.

Para la herramienta de fresado, se utiliza una llave ER, para aflojar la tuerca mostrada en la Figura 7-3, esto permitirá extraer el cono de presión y la fresa que hubiese en ese momento. Se realiza una limpieza al área y se coloca la herramienta requerida junto con el cono de presión adecuado. Por último, se aprieta la tuerca inicial.

Figura 7-3: Montaje herramienta de fresado

7.2 Fabricación

Se realiza la manufactura de un lote de seis mini implante dentales, cada uno de ellos, con características geométricas diferentes. A través del aplicativo para la obtención del código de producción (Ver Figura 6-3) se logra el siguiente programa:

```
%
#500=1.5 (Diametro roscado)
#501=7 (Longitud total)
#502=5.5 (Longitud roscada)
#503=0.3 (Paso)

M98 P20 (Fin de bloque)

#500=1.6 (Diametro roscado)
#501=6 (Longitud total)
#502=5 (Longitud no roscada)
#503=0.4 (Paso)

M98 P20 (Fin de bloque)

#500=1.7 (Diametro roscado)
#501=5 (Longitud total)
#502=4 (Longitud no roscada)
#503=0.3 (Paso)

M98 P20 (Fin de bloque)

#500=1.8 (Diametro roscado)
#501=7 (Longitud total)
#502=5 (Longitud no roscada)
#503=0.4 (Paso)

M98 P20 (Fin de bloque)

#500=1.9 (Diametro roscado)
#501=6 (Longitud total)
#502=4 (Longitud no roscada)
#503=0.3 (Paso)

M98 P20 (Fin de bloque)

#500=2 (Diametro roscado)
#501=7 (Longitud total)
#502=5.5 (Longitud no roscada)
#503=0.4 (Paso)

M98 P20 (Fin de bloque)

M30
%
```

El código anterior es enviado a través de un computador por el puerto de comunicación RS232 del controlador de la máquina. Para esta labor, es utilizado un *software* de transmisión de datos. Este procedimiento inicia con la adecuación del controlador para el recibimiento de información, se pulsa la tecla*DIAGN IN/OUT*seguido de la opción *INPUT,* por último se indica es espacio en donde se almacenará el programa y se pulsa *INPUT CALC*(Ver Figura 7-4).

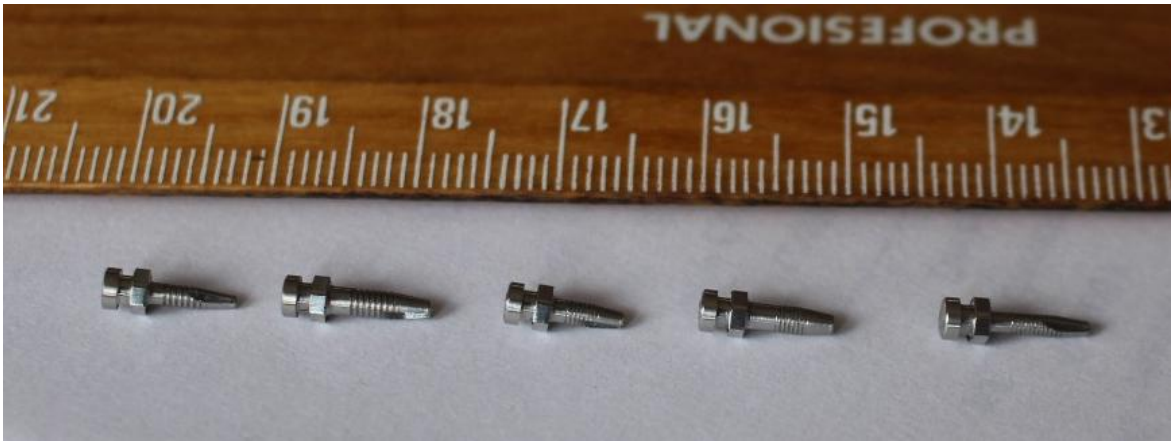
Figura 7-4: Procedimiento para la adecuación de recibimiento de información

Finalmente, se selecciona como activo al código de producción, para ello, se pulsa la tecla *MONITOR* en el controlador, seguido, se pulsa la opción SEARCH y se introduce el número del almacenamiento del programa de producción. El procedimiento se puede observar en la Figura 7-5. Teniendo seleccionado al programa de producción, se realiza una última inspección a la máquina y se inicia la producción.

Figura 7-5: Procedimiento de selección programa de producción

Una vez realizada la producción, se opbtiene el lote de mini implante mostrado en la Figura 7-6.

Figura 7-6: Lote de mini implantes fabricados.



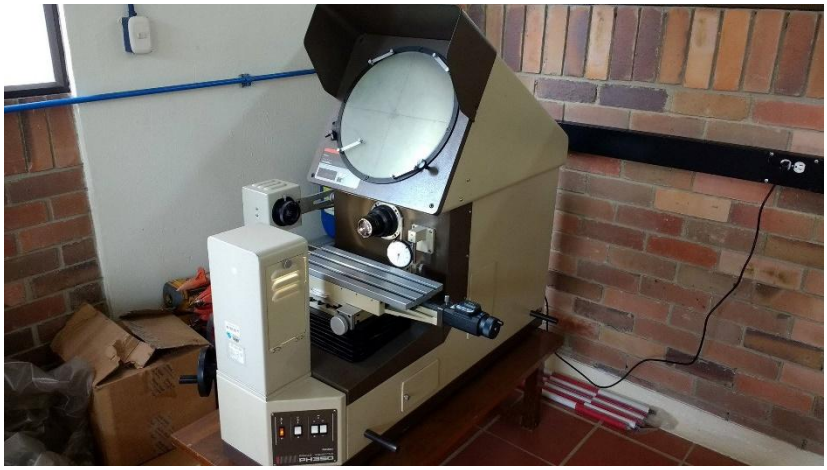
7.3 Metrología y compensación de herramientas

Se fabrica un primer lote de tres mini implantes dentales, que son sometidos a un control de calidad. Si bien, este tipo de implantes no requieren de gran precisión dimensional debido a que no serán anclados a componentes de alta precisión, por ello, desviaciones entre la micra y las 50 micras no afectan mayormente su funcionalidad, pero al ser un

componente de pequeñas magnitudes, los desfaces comunes (0.05 – 0.5 mm) en las medidas de procesos para piezas con un tamaño mas considerable, implica en este caso, variaciones a ser tenidas en cuenta. Por esto, al ser manufacturados en una maquina cuyas fabricaciones cotidianas no implican piezas mini, se debe realizar metrología a los productos y basados en estos resultados, se debe revisar las compensaciones en la posición de las herramientas, evitando los desfases comunes en piezas de tamaños mayores pero que en mini implantes representan un error.

La metrología a los implantes dentales se realizó en un proyector de perfiles marca Mitutoyo PH350 (Ver Figura 7-7). El cual se encuentra disponible en el laboratorio de metrología de la Universidad Santo Tomás.

Figura 7-7: Proyector de perfiles Mitutoyo PH350



Los mini implantes dentales son llevados al proyector de perfiles para obtener las diferentes medidas mostradas en la Figura 7-8. Los datos son registrados en la Tabla 7-1.

Figura 7-8: Plano de metrología

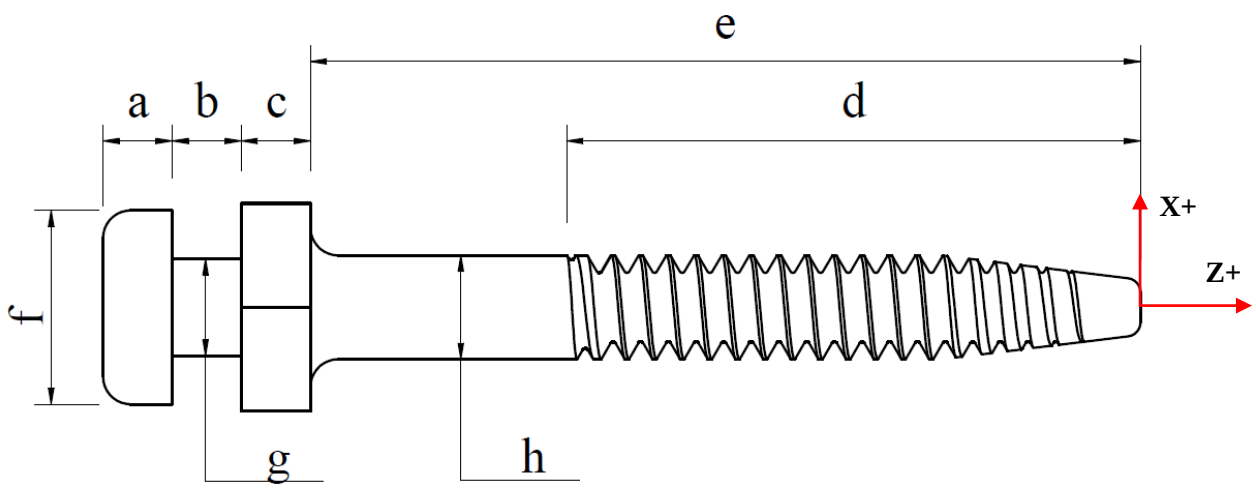


Tabla 7-1: Datos registrados de la metrología a un lote de 3 mini implantes

PROYECTOR DE PERFILES										
#	a	b	c	d	e	f	g	h	Paso	PF
1	1,10	0,50	1,07	3,88	6,04	3,134	2,190	1,652	0,39	0,290
2	1,08	0,46	1,08	5,38	7,00	3,112	2,166	1,866	0,39	0,290
3	1,08	0,50	1,08	4,89	6,45	3,149	2,211	2,146	0,30	0,358
Medidas de plano										
#	a	b	c	d	e	f	g	h	Paso	PF
1	1	0,5	1	4	6	3	2	1,5	0,4	
2	1	0,5	1	5,5	7	3	2	1,75	0,4	
3	1	0,5	1	5	6,5	3	2	2	0,3	
Diferencias										
#	a	b	c	d	e	f	g	h	Paso	PF
1	0,100	0,000	0,070	-0,120	0,040	0,134	0,190	0,152	-0,010	
2	0,080	-0,040	0,080	-0,120	0,000	0,112	0,166	0,116	-0,013	
3	0,080	0,000	0,080	-0,110	-0,050	0,149	0,211	0,146	0,000	
Prom	0,087	-0,013	0,077	-0,117	-0,003	0,132	0,189	0,138	-0,008	

La Tabla 7-1se encuentra dividida en tres bloques; el primero contiene las medidas, distribuidas de izquierda a derecha, de cada uno de los tres mini implantes; el segundo contiene las medidas reales que debe poseer la pieza fabricada; y por ultimo está la diferencia de las medidas de cada mini implante con respecto a la medida real. En la última fila se encuentra el promedio de las diferencias de cada medida. Al revisar los

valores obtenidos en la metrología, se comprueban que se tienen desviaciones que, para procesos de torneado, normalmente, no representan mayor importancia, pero en mini implantes se tienen que corregir.

Como se puede observar, las cotas *h* y *g*, que son diametrales, están desfasadas alrededor de una décima y mediapositiva, en el eje X; estas geometrías son elaboradas por las herramientas T0404 y T0606 respectivamente. Para cambiar esta compensación, nos dirigimos al controlador de la máquina, pulsamos la tecla *TOOL PARAM*(VerFigura 7-9),esto habilita la pantalla de compensaciones, se debe introducir el número de la herramienta que se desea compensar y la compensación en cada uno de los ejes. Es de tener en cuenta que si no se quiere modificar la compensación de un eje, solo basta con dejar el espacio vacío, además, las compensaciones introducidas están dadas en valores incrementales.

Además, debido a que la herramienta T0404 también maquina la geometría de la cota *f*, del eje X, este desfase presentado en esta cota, es corregido con la compensación ya realizada a dicha herramienta.

Figura 7-9: Procedimiento compensación herramental

En la cota d, igualmente se encuentra una diferencia de una décima en el eje Z. Se realizael mismo procedimiento para la compensación de la herramienta T0505 encargada

de realizar el roscado. En las diferencias encontradas de las cotas *a* y *c*se realiza una compensación de una décima negativa en el eje Z

8. Conclusiones y recomendaciones

8.1 Conclusiones

La cuarta revolución es una idea que cada vez coge más fuerza, es por eso que la academia nacional debe empezar a preparar a sus estudiantes para los futuros retos. Este tipo de investigaciones, donde se trata de imprimir cierta autonomía al proceso, junto con el enfoque social incluido, son de gran importancia para los próximos retos, en términos de industria, que enfrentará el país.

Se ha demostrado las bondades y a los niveles de flexibilidad, a los que se puede llegar con la automatización de *software* CAD, a través, de la programación de API y el uso de herramientas del *Kernel*. La extrapolación a necesidades reales de la industria colombiana, como se realizó en este trabajo, ayuda en la aceleración y optimización de procesos de ingeniería.

La programación paramétrica embebida en los controladores CNC y a través de subprogramas, junto con aplicativos desarrollados en plataformas de programación, dieron como resultado un proceso de fabricación ágil y flexible, de familias de mini-implantes en función de sus principales características geométricas. Esto señaló que, con un buen análisis, y sabiendo utilizar el máximo potencial y capacidades de la máquina, es posible obtener grandes beneficios con poca inversión monetaria.

Los tiempos de producción fueron reducidos en 2.67 min, representando una reducción del 34.1% al eliminar el cambio de parámetros de forma manual, el cual era equivalente al 51.7% del tiempo de maquinado.Además, la metrología realizada en un proyector de perfiles fue suficiente para realizar las compensaciones en la posición de las herramientas, reduciendo el valor de error. Sin embargo, y pos las dimensiones del roscado, se requiere de instrumentos de medición con mayor presicion para la metrología de esta operación.

8.2 Recomendaciones

Se recomienda a aquel lector, que esté interesado en explorar más a fondo la programación de API's, para la automatización de procesos de dibujo en Solid Edge, leer la guía SDK (*Software Development Kit*), que se encuentra en la carpeta de instalación del *software*. En esta guía, encontrará toda la información necesaria sobre el uso de los métodos del *kernel*, y en algunos casos se incluye algún ejemplo.

Bibliografía

[1] M. Brettel, M. Klein, and N. Friederichsen, “The Relevance of Manufacturing Flexibility in the Context of Industrie 4.0,” *Procedia CIRP*, vol. 41, pp. 105–110, 2016.

[2] R. Beach, A. P. Muhlemann, D. H. R. Price, A. Paterson, and J. A. Sharp, “Review of manufacturing flexibility,” *Eur. J. Oper. Res.*, vol. 122, no. 1, pp. 41–57, 2000.

[3] M. Garzon, “Micromanufactura: retos gigantes,” *Met. Mec.*, vol. 20, pp. 12–17, 2015.

[4] K. B. Kim and P. H. Buschang, “Considerations for Optimizing the Use of Miniscrew Implants in Orthodontic Practice,” in *Temporary skeletal Anchorage Devices*, K. B. Kim, Ed. Berlin: Springer-Verlag Berlin Heidelberg, 2014, pp. 1–27.

[5] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, “Industrie 4.0,” *WIRTSCHAFTSINFORMATIK*, vol. 56, no. 4, pp. 261–264, 2014.

[6] H. M. Deitel and P. J. Deitel, *Cómo programar en C++*, Cuarta. México D.F.: Pearson Educaci{ó}n, 2003.

[7] C. Becerra, *C++ Una herramienta para la programación orientada a objetos*, Cuarta. Bogotá: Kimpres, 1999.

[8] H. M. Deitel, P. J. Deitel, and A. V. R. Elizondo, *Cómo programar en Java*, Séptima. México D.F.: Pearson Educación, 2008.

[9] K. E. Hernandez, E. Olivares, and C. A. Zuñiga, “Safety stock levels in modular product system using commonality and part families,” *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 1387–1392, 2015.

[10] R. Sturm, C. Pollard, and J. Craig, “Chapter 11 - Application Programming Interfaces and Connected Systems,” in *Application Performance Management*

(APM) in the Digital Enterprise, R. Sturm, C. Pollard, and J. Craig, Eds. Boston: Morgan Kaufmann, 2017, pp. 137–150.

[11] P. Nyhuis, N. Duffie, M. Brieke, and I. Management, "Changeable Manufacturing - Classification , Design and Operation," *CIRP Ann. - Manuf. Technol.*, vol. 56, no. 2, pp. 783–809, 2007.

[12] B. J. Browne, D. Dubois, K. Rathmill, S. P. Sethi, and K. E. Stecke, "Classification of flexible manufacturing systems," *FMS Mag.*, pp. 114–117, 1984.

[13] I. S. Kang, J. S. Kim, J. H. Kim, M. C. Kang, and Y. W. Seo, "A mechanistic model of cutting force in the micro end milling process," *J. Mater. Process. Technol.*, vol. 188, pp. 250–255, 2007.

[14] M. P. Groover, *Fundamentos de manufactura moderna*, Tercera Ed. México D.F.: Mc Graw-Hill, 2007.

[15] G. Kiswanto, D. L. Zariatin, and T. J. Ko, "The effect of spindle speed, feed-rate and machining time to the surface roughness and burr formation of Aluminum Alloy 1100 in micro-milling operation," *J. Manuf. Process.*, vol. 16, pp. 435–450, 2014.

[16] J. B. Cope, "Temporary anchorage devices in orthodontics: A paradigm shift," *Semin. Orthod.*, vol. 11, no. 1 SPEC. ISS., pp. 3–9, 2005.

[17] B. Melsen, "Mini-Implants : Where Are We?," *J. Clin. Orthod.*, vol. XXXIX, no. 9, pp. 539–547, 2005.

[18] D. M. Brunette, P. Tengvall, M. Textor, and P. Thomsen, *Titanium in Medicine: Material Science, Surface Science, Engineering, Biological Responses and Medical Applications*, Ilustrated. Springer Science & Business Media, 2012.

[19] M. Migliorati, A. Signori, and A. Silvestrini Biavati, "Temporary anchorage device stability: An evaluation of thread shape factor," *Eur. J. Orthod.*, vol. 34, no. 5, pp. 582–586, 2012.

[20] J. A. García Barbosa, J. M. Arroyo Osorio, and E. Córdoba Nieto, "Simulation and

verification of parametric numerical control programs using a virtual machine tool," *Prod. Eng.*, vol. 8, no. 3, pp. 407–413, 2014.

[21] M. Djassemi, "A Parametric Programming Technique For Efficient CNC Machining Operations," *Comput. Ind. Eng.*, vol. 35, no. 98, pp. 33–36, 1998.

[22] Y. Bodein, B. Rose, and E. Caillaud, "Computers in Industry Explicit reference modeling methodology in parametric CAD system," *Comput. Ind.*, vol. 65, no. 1, pp. 136–147, 2014.

A. Anexo: Código Fuente API Modelamiento Tridimensional

```
Imports SolidEdgeConstants

Public Class Implante
    Private Dia As Double
    Private Lon As Double
    Private LonR As Double
    Private P As Double
    Private TR As Boolean
    Private AI As Double
    Private ASH As Double
    Private Perfil As SolidEdgePart.Profile
    Private PerfilHex As SolidEdgePart.Profile
    Private PerfilRosca As SolidEdgePart.Profile
    Private PerfilFilo As SolidEdgePart.Profile
    Private Lineas2d As SolidEdgeFrameworkSupport.Lines2d
    Private Arcos2d As SolidEdgeFrameworkSupport.Arcs2d
Private Modelo As SolidEdgePart.Model

    'Constructores

    Public Sub New()
        Me.Diametro = 1.6 / 1000
        Me.Longitud = 12 / 1000
        Me.LongRos = 8 / 1000
        Me.Paso = 0.5 / 1000
        Me.TipoRosca = False 'Verdadero: Metrica, Falso: Sierra
Me.AngIncli = 5 * Math.PI / 180
        Me.AngSal = 15 * Math.PI / 180
        Me.Profile = Nothing
        Me.ProfileHex = Nothing
        Me.ProfileRosca = Nothing
        Me.ProfileFilo = Nothing
        Me.Lines2d = Nothing
        Me.Arcs2d = Nothing
        Me.Model = Nothing
    End Sub
```

```
    Public Sub New(Diametro As Double, Longitud As Double, LongRos
As Double, Paso As Double, TipoRosca As Boolean, AngIncli As
Double, AngSal As Double)
Me.Diametro = Diametro / 1000
        Me.Longitud = Longitud / 1000
        Me.LongRos = LongRos / 1000
        Me.Paso = Paso / 1000
        Me.TipoRosca = TipoRosca 'Verdadero: Metrica, Falso:
Sierra
Me.AngIncli = AngIncli * Math.PI / 180
        Me.AngSal = AngSal * Math.PI / 180
        Me.Profile = Nothing
        Me.ProfileHex = Nothing
        Me.ProfileRosca = Nothing
        Me.Lines2d = Nothing
        Me.Arcs2d = Nothing
    End Sub

    'Set's Get's

    Public Property Diametro() As Double
        Get
            Return Me.Dia
        End Get
        Set(ByVal Value As Double)
            Me.Dia = Value
        End Set
    End Property

    Public Property Longitud() As Double
        Get
            Return Me.Lon
        End Get
        Set(ByVal Value As Double)
            Me.Lon = Value
        End Set
    End Property

    Public Property LongRos() As Double
        Get
            Return Me.LonR
        End Get
        Set(ByVal Value As Double)
            Me.LonR = Value
        End Set
    End Property

    Public Property Paso() As Double
        Get
            Return Me.P
```

```
        End Get
        Set(ByVal Value As Double)
            Me.P = Value
        End Set
    End Property

    Public Property TipoRosca() As Boolean
        Get
            Return Me.TR
        End Get
        Set(ByVal Value As Boolean)
            Me.TR = Value
        End Set
    End Property

    Public Property AngIncli() As Double
        Get
            Return Me.AI
        End Get
        Set(ByVal Value As Double)
            Me.AI = Value
        End Set
    End Property

    Public Property AngSal() As Double
        Get
            Return Me.ASH
        End Get
        Set(ByVal Value As Double)
            Me.ASH = Value
        End Set
    End Property

    Public Property Profile() As SolidEdgePart.Profile
        Get
            Return Me.Perfil
        End Get
        Set(ByVal Value As SolidEdgePart.Profile)
            Me.Perfil = Value
        End Set
    End Property

    Public Property ProfileHex() As SolidEdgePart.Profile
        Get
            Return Me.PerfilHex
        End Get
        Set(ByVal Value As SolidEdgePart.Profile)
            Me.PerfilHex = Value
        End Set
    End Property
```

```
Public Property ProfileRosca() As SolidEdgePart.Profile
    Get
        Return Me.PerfilRosca
    End Get
    Set(ByVal Value As SolidEdgePart.Profile)
        Me.PerfilRosca = Value
    End Set
End Property

Public Property ProfileFilo() As SolidEdgePart.Profile
    Get
        Return Me.PerfilFilo
    End Get
    Set(ByVal Value As SolidEdgePart.Profile)
        Me.PerfilFilo = Value
    End Set
End Property

Public Property Lines2d() As SolidEdgeFrameworkSupport.Lines2d
    Get
        Return Me.Lineas2d
    End Get
    Set(ByVal Value As SolidEdgeFrameworkSupport.Lines2d)
        Me.Lineas2d = Value
    End Set
End Property

Public Property Arcs2d() As SolidEdgeFrameworkSupport.Arcs2d
    Get
        Return Me.Arcos2d
    End Get
    Set(ByVal Value As SolidEdgeFrameworkSupport.Arcs2d)
        Me.Arcos2d = Value
    End Set
End Property

Public Property Model() As SolidEdgePart.Model
    Get
        Return Me.Modelo
    End Get
    Set(ByVal Value As SolidEdgePart.Model)
        Me.Modelo = Value
    End Set
End Property

'toString

Overrides Function ToString() As String
```

```
        Return "Diametro: " + Diametro.ToString & vbCrLf &
"Longitud: " + Longitud.ToString & vbCrLf &
"Longitud Roscada: " + LongRos.ToString & vbCrLf & "Paso: " +
Paso.ToString & vbCrLf & "Tipo de Rosca: " + TipoRosca.ToString &
vbCrLf &
            "Angulo de Inclinación: " + AngIncli.ToString & vbCrLf
& "Angulo de Salida: " + AngSal.ToString
End Function

    'Funciones con logica

    Public Function CreacionPlano(Part As
SolidEdgePart.PartDocument, Perfil As SolidEdgePart.Profile,
NumPlane As Integer, Optional Dis As Double = 0) As
SolidEdgePart.Profile
        Dim ProfileSets As SolidEdgePart.ProfileSets =
Part.ProfileSets
        Dim ProfileSet As SolidEdgePart.ProfileSet =
ProfileSets.Add()
        Dim Profiles As SolidEdgePart.Profiles =
ProfileSet.Profiles
        Dim RefPlanes As SolidEdgePart.RefPlanes = Part.RefPlanes

        If Dis <> 0 Then
            Perfil =
Profiles.Add(RefPlanes.AddParallelByDistance(RefPlanes.Item(NumPla
ne), Dis, SolidEdgePart.ReferenceElementConstants.igNormalSide))
        Else
            Perfil = Profiles.Add(RefPlanes.Item(NumPlane))
        End If
        Return Perfil
    End Function

    Public Function CreacionPlanoAngulo(Part As
SolidEdgePart.PartDocument, Perfil As SolidEdgePart.Profile,
NumPlane As Integer, Dis As Double, Ang As Double) As
SolidEdgePart.Profile
        Dim ProfileSets As SolidEdgePart.ProfileSets =
Part.ProfileSets
        Dim ProfileSet As SolidEdgePart.ProfileSet =
ProfileSets.Add()
        Dim Profiles As SolidEdgePart.Profiles =
ProfileSet.Profiles
        Dim RefPlanes As SolidEdgePart.RefPlanes = Part.RefPlanes

        Perfil =
Profiles.Add(RefPlanes.AddAngularByAngle(RefPlanes.AddParallelByDi
stance(RefPlanes.Item(NumPlane), Dis,
SolidEdgePart.ReferenceElementConstants.igNormalSide), Ang *
Math.PI / 180,
```

```
SolidEdgePart.ReferenceElementConstants.igNormalSide,
RefPlanes.Item(1),
SolidEdgePart.ReferenceElementConstants.igPivotEnd))

        Return Perfil
    End Function

    Public Sub PerfilImplante(Part As SolidEdgePart.PartDocument,
Perfil As SolidEdgePart.Profile)
        Me.Profile = Perfil
        Me.Lines2d = Me.Profile.Lines2d
        Dim Line2d As SolidEdgeFrameworkSupport.Line2d = Nothing
        Me.Arcs2d = Me.Profile.Arcs2d
        Dim Arc2d As SolidEdgeFrameworkSupport.Arc2d = Nothing

        Dim R As Double = Me.Diametro / 2
        Dim C04 As Double = 0.4 / 1000 : Dim C07 As Double = 0.7 /
1000 : Dim C1 As Double = 1 / 1000
        Dim C14 As Double = 1.4 / 1000 : Dim C3 As Double = 3 /
1000 : Dim C34 As Double = 3.4 / 1000
        Dim C05 As Double = R - Pf() - (0.1 / 1000)

        Line2d = Me.Lines2d.AddBy2Points(0, 0, 0, C14)
        Line2d = Me.Lines2d.AddBy2Points(0, C14, C1, C14)
        Line2d = Me.Lines2d.AddBy2Points(C1, C14, C1, C07)
        Line2d = Me.Lines2d.AddBy2Points(C1, C07, C3, C07)
        Line2d = Me.Lines2d.AddBy2Points(C3, C07, C3, R + C04)
        Arc2d = Me.Arcs2d.AddByCenterStartEnd(C34, R + C04, C3, R
+ C04, C34, R)
        Line2d = Me.Lines2d.AddBy2Points(C34, R, Me.Longitud + C3,
R)
        Line2d = Me.Lines2d.AddBy2Points(Me.Longitud + C3, 0,
Me.Longitud + C3, C05)
        Line2d = Me.Lines2d.AddByPointAngleLength(Me.Longitud +
C3, C05, 173 * Math.PI / 180, Me.LongRos)
        Me.Lineas2d.Item(8).Trim(0, C3, Me.Lineas2d.Item(6))
        Me.Lineas2d.Item(6).Trim(Me.Longitud + C3, 0,
Me.Lineas2d.Item(8))
        Line2d = Me.Lines2d.AddBy2Points(Me.Longitud + C3, 0, 0,
0)
        Arc2d = Arcs2d.AddAsFillet(Me.Lines2d.Item(1),
Me.Lines2d.Item(2), 0.4 / 1000, 0.4 / 1000, 0.4 / 1000) 'Redondeo
Cabeza
        Arc2d = Arcs2d.AddAsFillet(Me.Lines2d.Item(7),
Me.Lines2d.Item(8), 0.2 / 1000, 0, 0) 'Redondeo final

If CerrarPerfilImplante() <> 0 Then
            MsgBox("Perfil para la base de la revolución no se
cerro corectamente, verifique sus datos de entrada e intentelo de
```

```
nuevo ", MsgBoxStyle.Critical Or MsgBoxStyle.SystemModal, "Error
Critico")
Application.Exit()
            Part.Close()
        End If
        Me.Profile.Visible = False
        Me.Lines2d = Nothing
        Me.Arcs2d = Nothing
    End Sub

    Public Function CerrarPerfilImplante() As
SolidEdgePart.ProfileValidationType
        Dim Relations2d As SolidEdgeFrameworkSupport.Relations2d =
Me.Profile.Relations2d
        Dim Relation2d As SolidEdgeFrameworkSupport.Relation2d =
Nothing

        'Cierre del perfil
        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(2),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(3),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(3),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(4),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(4),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(5),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(5),
KeypointIndexConstants.igLineEnd, Me.Arcs2d.Item(1),
KeypointIndexConstants.igArcStart)
        Relation2d = Relations2d.AddKeypoint(Me.Arcs2d.Item(1),
KeypointIndexConstants.igArcEnd, Me.Lines2d.Item(6),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(7),
KeypointIndexConstants.igLineStart, Me.Lines2d.Item(9),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(9),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(1),
KeypointIndexConstants.igLineStart)
        Return
Me.Profile.End(SolidEdgePart.ProfileValidationType.igProfileClosed
)
    End Function

    Public Function EjeRevolucion(index As Integer, Perfil As
SolidEdgePart.Profile) As SolidEdgePart.RefAxis
        Return
Perfil.SetAxisOfRevolution(Perfil.Lines2d.Item(index))
    End Function
```

```
    Public Sub Revolucion(Part As SolidEdgePart.PartDocument, Ang
As Double)
        Dim Models As SolidEdgePart.Models = Part.Models
        Me.Model = Nothing
        Dim Array(0 To 1) As SolidEdgePart.Profile

        Array(0) = Me.Profile
        Me.Model = Models.AddFiniteRevolvedProtrusion(1, Array,
EjeRevolucion(9, Me.Profile),
SolidEdgePart.FeaturePropertyConstants.igRight, Ang * Math.PI /
180)
    End Sub

    Public Function Pf() As Double
        Dim ProF As Double = Nothing
        If Me.TipoRosca Then
            ProF = ((((7 * Math.Sqrt(3)) / 16) * Me.Paso) - (0.05
/ 1000))
        Else
            Dim Ang1 As Double = (60 * Math.PI) / 180
            Dim Ang2 As Double = ((30 * Math.PI) / 180) +
Me.AngIncli
            ProF = (((7 / 8) * Me.Paso) / (Math.Sin(Ang1))) *
Math.Sin(Ang2) * Math.Cos(Me.AngIncli) - (0.0406308 / 1000)
        End If
        Return ProF
    End Function

    Public Sub PerfilHexagono(Part As SolidEdgePart.PartDocument,
Perfil As SolidEdgePart.Profile)
        Me.ProfileHex = Perfil
        Me.Lines2d = Me.ProfileHex.Lines2d
        Dim Line2d As SolidEdgeFrameworkSupport.Line2d = Nothing

        Lines2d = Me.ProfileHex.Lines2d
        Line2d = Me.Lines2d.AddBy2Points(-0.865 / 1000, 1.5 /
1000, 0.865 / 1000, 1.5 / 1000)
        Line2d = Me.Lines2d.AddBy2Points(0.865 / 1000, 1.5 / 1000,
1.73 / 1000, 0)
        Line2d = Me.Lines2d.AddBy2Points(1.73 / 1000, 0, 0.865 /
1000, -1.5 / 1000)
        Line2d = Me.Lines2d.AddBy2Points(0.865 / 1000, -1.5 /
1000, -0.865 / 1000, -1.5 / 1000)
        Line2d = Me.Lines2d.AddBy2Points(-0.865 / 1000, -1.5 /
1000, -1.73 / 1000, 0)
        Line2d = Me.Lines2d.AddBy2Points(-1.73 / 1000, 0, -0.865 /
1000, 1.5 / 1000)

        If CerrarPerfilHexagono() <> 0 Then
```

```
MsgBox("Perfil abierto, verifique las dimensiones introducidas",
MsgBoxStyle.Critical Or MsgBoxStyle.SystemModal, "Error Critico")
Application.Exit()
            Part.Close()
        End If
        ProfileHex.Visible = False
        Me.Lines2d = Nothing
        Line2d = Nothing
    End Sub

    Public Function CerrarPerfilHexagono() As
SolidEdgePart.ProfileValidationType
        Dim Relations2d As SolidEdgeFrameworkSupport.Relations2d =
Me.Profile.Relations2d
        Dim Relation2d As SolidEdgeFrameworkSupport.Relation2d =
Nothing

        For i As Integer = 1 To 6 Step 1
            If i <> 6 Then
                Relation2d =
Relations2d.AddKeypoint(Me.Lines2d.Item(i),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(i + 1),
KeypointIndexConstants.igLineStart)
            ElseIf i = 6 Then
                Relation2d =
Relations2d.AddKeypoint(Me.Lines2d.Item(i),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(1),
KeypointIndexConstants.igLineStart)
            End If
        Next i
        Return
Me.Profile.End(SolidEdgePart.ProfileValidationType.igProfileClosed
)
    End Function

    Public Sub ExtrusionHex(Part As SolidEdgePart.PartDocument,
Dis As Double)
        Dim Models As SolidEdgePart.Models = Part.Models
        Dim Model As SolidEdgePart.Model = Nothing
        Dim Array(0 To 1) As SolidEdgePart.Profile

        Array(0) = Me.ProfileHex
        Model = Models.AddFiniteExtrudedProtrusion(1, Array,
SolidEdgePart.FeaturePropertyConstants.igRight, Dis)
    End Sub

    Public Sub Helice(Part As SolidEdgePart.PartDocument,
TipoRosca As Boolean, Perfil As SolidEdgePart.Profile)
        Me.ProfileRosca = Perfil
        Dim HelixCutout As SolidEdgePart.HelixCutout = Nothing
```

```
        Dim Array As Array = Nothing

        Array =
Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
        If TipoRosca Then
            Array.SetValue(PerfilMetrico(Part, Me.ProfileRosca),
0)
        Else
            Array.SetValue(PerfilSierra(Part, Me.ProfileRosca), 0)
        End If
        HelixCutout =
Part.Models(0).HelixCutouts.AddFinite(EjeRevolucion(4,
Me.ProfileRosca), SolidEdgePart.FeaturePropertyConstants.igStart,
1,

Array, SolidEdgePart.FeaturePropertyConstants.igRight, 0.1,
Me.Paso, Me.LongRos / Me.Paso,

SolidEdgePart.FeaturePropertyConstants.igRight)
    End Sub

    Public Sub Helice2(Part As SolidEdgePart.PartDocument,
TipoRosca As Boolean, Perfil As SolidEdgePart.Profile)
        Me.ProfileRosca = Perfil
        Dim HelixCutout As SolidEdgePart.HelixCutout = Nothing
        Dim Array As Array = Nothing

        Array =
Array.CreateInstance(GetType(SolidEdgePart.Profile), 1)
        If TipoRosca Then
            Array.SetValue(PerfilMetrico(Part, Me.ProfileRosca),
0)
        Else
            Array.SetValue(PerfilSierra(Part, Me.ProfileRosca), 0)
        End If

        HelixCutout =
Part.Models(0).HelixCutouts.AddFinite(EjeRevolucion(4,
Me.ProfileRosca), SolidEdgePart.FeaturePropertyConstants.igEnd, 1,

Array, SolidEdgePart.FeaturePropertyConstants.igRight, 0.1,
Me.Paso, 2,

SolidEdgePart.FeaturePropertyConstants.igRight, -Me.AngSal)
    End Sub

    Public Function PerfilMetrico(Part As
SolidEdgePart.PartDocument, Perfil As SolidEdgePart.Profile) As
SolidEdgePart.Profile
        Me.ProfileRosca = Perfil
```

```
        Me.Lines2d = Me.ProfileRosca.Lines2d
        Dim Line2d As SolidEdgeFrameworkSupport.Line2d = Nothing
        Me.Arcs2d = Me.ProfileRosca.Arcs2d
        Dim Arc2d As SolidEdgeFrameworkSupport.Arc2d = Nothing

        Dim H = (Math.Sqrt(3) / 2) * Me.Paso
        Dim x1 = (3 / 1000) + (Me.Longitud - Me.LongRos)
        Dim x2 = x1 + (Me.Paso - ((2 * H) / (Math.Sqrt(3) * 8)))
        Dim y = Me.Diametro / 2
        Dim Lon As Double = ((H - (H / 8)) / Math.Cos(30 * Math.PI
/ 180))

        Line2d = Me.Lines2d.AddBy2Points(x1, y, x2, y)
        Line2d = Me.Lines2d.AddByPointAngleLength(x1, y, -(60 *
Math.PI) / 180, Lon)
        Line2d = Me.Lines2d.AddByPointAngleLength(x2, y, -(120 *
Math.PI) / 180, Lon)
        Line2d = Me.Lines2d.AddBy2Points(0, 0, (3 / 1000) +
Me.Longitud, 0)
        Arc2d = Me.Arcs2d.AddAsFillet(Me.Lines2d.Item(2),
Me.Lines2d.Item(3), 0.05 / 1000, x1, Me.Diametro)

        Me.ProfileRosca.Visible = False
        Return Me.ProfileRosca
        Me.Lines2d = Nothing
        Me.Arcs2d = Nothing
    End Function

    Public Function PerfilSierra(Part As
SolidEdgePart.PartDocument, Perfil As SolidEdgePart.Profile) As
SolidEdgePart.Profile
        Me.ProfileRosca = Perfil
        Me.Lines2d = Me.ProfileRosca.Lines2d
        Dim Line2d As SolidEdgeFrameworkSupport.Line2d = Nothing
        Me.Arcs2d = Me.ProfileRosca.Arcs2d
        Dim Arc2d As SolidEdgeFrameworkSupport.Arc2d

Dim x1 = (3 / 1000) + (Me.Longitud - Me.LongRos)
        Dim x2 = x1 + (7 * Me.Paso / 8)
        Dim y = Me.Diametro / 2
        Dim Ang1 As Double = (60 * Math.PI) / 180 'Angulo del
inserto, va en relación con el paso por ley de senos
        Dim Ang2 As Double = ((30 * Math.PI) / 180) + Me.AngIncli
'Angulo izquierdo del perfil
        Dim Ang3 As Double = ((90 * Math.PI) / 180) + Me.AngIncli
' Angulo derecho del perfil
        Dim Lon1 As Double = 7 * Me.Paso / 8 ' es igual al paso -
paso/8
        Dim Lon2 As Double = (Lon1 * Math.Sin(Ang2)) /
Math.Sin(Ang1) ' Longitud derecha del perfil
```

```
        Dim Lon3 As Double = (Lon1 * Math.Sin(Ang3)) /
Math.Sin(Ang1) ' Longitud izquierda del perfil

Line2d = Me.Lines2d.AddByPointAngleLength(x1, y, 0, Lon1)
        Line2d = Me.Lines2d.AddByPointAngleLength(x1, y, -Ang2,
Lon2)
        Line2d = Me.Lines2d.AddByPointAngleLength(x2, y, -Ang3,
Lon3)
        Line2d = Me.Lines2d.AddBy2Points(0, 0, (3 / 1000) +
Me.Longitud, 0)
        Arc2d = Me.Arcs2d.AddAsFillet(Me.Lines2d.Item(2),
Me.Lines2d.Item(3), 0.05 / 1000, x1, Me.Diametro)

        Me.ProfileRosca.Visible = False
        Return Me.ProfileRosca
        Me.Lines2d = Nothing
        Me.Arcs2d = Nothing
    End Function

    Public Sub PerfilFilos(Part As SolidEdgePart.PartDocument,
Perfil As SolidEdgePart.Profile, Optional i As Double = 1)
        Me.ProfileFilo = Perfil
        Me.Lines2d = Me.ProfileFilo.Lines2d
        Dim Line2d As SolidEdgeFrameworkSupport.Line2d = Nothing
        Me.Arcs2d = Me.ProfileFilo.Arcs2d
        Dim Arc2d As SolidEdgeFrameworkSupport.Arc2d = Nothing
        Dim Circles2d As SolidEdgeFrameworkSupport.Circles2d =
Me.ProfileFilo.Circles2d
        Dim Circle2d As SolidEdgeFrameworkSupport.Circle2d =
Nothing

        Dim C3 As Double = 3 / 1000 : Dim C2 As Double = 2 / 1000
: Dim C1 As Double = 1 / 1000
        Dim C25 As Double = 2.5 / 1000 : Dim C15 As Double = 1.5 /
1000 : Dim C45 As Double = 4.5 / 1000
        Dim CI As Double = (Me.Diametro / 2) - Pf() - (0.276945 /
1000)

        Me.Lines2d = Me.ProfileFilo.Lines2d
        Line2d = Me.Lines2d.AddByPointAngleLength(Me.Longitud +
C3, i * CI, Math.PI, C2)
        Line2d = Me.Lines2d.AddByPointAngleLength(Me.Longitud -
C15, i * (C25 + CI), 0, i * C45)
        Line2d = Me.Lines2d.AddByPointAngleLength(Me.Longitud +
C3, i * (C25 + CI), -Math.PI / 2, i * C25)
        Circle2d = Circles2d.AddByCenterRadius(Me.Longitud + C1, i
* (CI + C25), i * C25)
        Circles2d.Item(1).Trim(1, i * 1, Me.Lines2d.Item(2),
Me.Lines2d.Item(1))
```

```
If CerrarPerfilFilos() Then '<> 0 Then
            MsgBox("Perfil abierto, verifique las dimensiones
introducidas", MsgBoxStyle.Critical Or MsgBoxStyle.SystemModal,
"Error Critico")
Application.Exit()
            Part.Close()
        End If

        ProfileFilo.Visible = False
        Me.Lines2d = Nothing
    End Sub

    Public Function CerrarPerfilFilos() As
SolidEdgePart.ProfileValidationType
        Dim Relations2d As SolidEdgeFrameworkSupport.Relations2d =
Me.ProfileFilo.Relations2d
        Dim Relation2d As SolidEdgeFrameworkSupport.Relation2d =
Nothing

        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(1),
KeypointIndexConstants.igLineStart, Me.Lines2d.Item(3),
KeypointIndexConstants.igLineEnd)
        Relation2d = Relations2d.AddKeypoint(Me.Lines2d.Item(3),
KeypointIndexConstants.igLineStart, Me.Lines2d.Item(2),
KeypointIndexConstants.igLineEnd)

        Return
Me.ProfileFilo.End(SolidEdgePart.ProfileValidationType.igProfileCl
osed)
    End Function

    Public Sub ExtrusionFilo(Part As SolidEdgePart.PartDocument,
Dis As Double, Lado As SolidEdgePart.FeaturePropertyConstants)
        Dim ExtCut As SolidEdgePart.ExtrudedCutout

        ExtCut = Me.Model.ExtrudedCutouts.AddFinite(ProfileFilo,
Lado, Lado, Dis)
    End Sub

End Class
```

```
Imports SolidEdgeConstants

Public Class Hueso
    Private An As Double
    Private Al As Double
    Private Es As Double
    Private DNicho As Double
    Private TR As Boolean
```

```
    Private ASH As Double
    Private Perfil As SolidEdgePart.Profile
    Private Perfil2 As SolidEdgePart.Profile
    Private Lineas As SolidEdgeFrameworkSupport.Lines2d
    Private Circulos As SolidEdgeFrameworkSupport.Circles2d
    Private Model As SolidEdgePart.Model

    Public Sub New()
        Me.Ancho = 5 / 1000
Me.Alto = 5 / 1000
        Me.Espesor = 3 / 1000
        Me.DiametroNicho = 1.5 / 1000
        Me.TipoRosca = True 'Verdadero: Metrica, Falso: Sierra
Me.AngSal = 15 * Math.PI / 180
        Me.Profile = Nothing
        Me.ProfileCut = Nothing
        Me.Lines2d = Nothing
        Me.Circles2d = Nothing
    End Sub

    Public Sub New(An As Double, Al As Double, Es As Double,
DNicho As Double, TR As Boolean, ASH As Double)
Me.Ancho = An / 1000
        Me.Alto = Al / 1000
        Me.Espesor = Es / 1000
        Me.DiametroNicho = DNicho / 1000
        Me.TipoRosca = TR 'Verdadero: Metrica, Falso: Sierra
Me.AngSal = ASH * Math.PI / 180
        Me.Profile = Nothing
        Me.ProfileCut = Nothing
        Me.Lines2d = Nothing
        Me.Circles2d = Nothing
    End Sub

    Public Property Ancho() As Double
        Get
            Return Me.An
        End Get
        Set(ByVal Value As Double)
            Me.An = Value
        End Set
    End Property

    Public Property Alto() As Double
        Get
            Return Me.Al
        End Get
        Set(ByVal Value As Double)
            Me.Al = Value
        End Set
```

```
End Property

Public Property Espesor() As Double
    Get
        Return Me.Es
    End Get
    Set(ByVal Value As Double)
        Me.Es = Value
    End Set
End Property

Public Property DiametroNicho() As Double
    Get
        Return Me.DNicho
    End Get
    Set(ByVal Value As Double)
        Me.DNicho = Value
    End Set
End Property

Public Property TipoRosca() As Boolean
    Get
        Return Me.TR
    End Get
    Set(ByVal Value As Boolean)
        Me.TR = Value
    End Set
End Property

Public Property AngSal() As Double
    Get
        Return Me.ASH
    End Get
    Set(ByVal Value As Double)
        Me.ASH = Value
    End Set
End Property

Public Property Profile() As SolidEdgePart.Profile
    Get
        Return Me.Perfil
    End Get
    Set(ByVal Value As SolidEdgePart.Profile)
        Me.Perfil = Value
    End Set
End Property

Public Property ProfileCut() As SolidEdgePart.Profile
    Get
        Return Me.Perfil2
```

```
        End Get
        Set(ByVal Value As SolidEdgePart.Profile)
            Me.Perfil2 = Value
        End Set
    End Property

    Public Property Lines2d() As SolidEdgeFrameworkSupport.Lines2d
        Get
            Return Me.Lineas
        End Get
        Set(ByVal Value As SolidEdgeFrameworkSupport.Lines2d)
            Me.Lineas = Value
        End Set
    End Property

    Public Property Circles2d() As
SolidEdgeFrameworkSupport.Circles2d
        Get
            Return Me.Circulos
        End Get
        Set(ByVal Value As SolidEdgeFrameworkSupport.Circles2d)
            Me.Circulos = Value
        End Set
    End Property

    'Funciones con logica

    Public Sub PerfilHueso(Part As SolidEdgePart.PartDocument,
Perfil As SolidEdgePart.Profile)
        Me.Profile = Perfil
        Me.Lines2d = Me.Profile.Lines2d
        Dim Line2d As SolidEdgeFrameworkSupport.Line2d = Nothing


        Line2d = Me.Lines2d.AddBy2Points(Me.Ancho / 2, Me.Alto /
2, Me.Ancho / 2, -Me.Alto / 2)
        Line2d = Me.Lines2d.AddBy2Points(Me.Ancho / 2, -Me.Alto /
2, -Me.Ancho / 2, -Me.Alto / 2)
        Line2d = Me.Lines2d.AddBy2Points(-Me.Ancho / 2, -Me.Alto /
2, -Me.Ancho / 2, Me.Alto / 2)
        Line2d = Me.Lines2d.AddBy2Points(-Me.Ancho / 2, Me.Alto /
2, Me.Ancho / 2, Me.Alto / 2)

If CerrarPerfil() <> 0 Then
            MsgBox("Perfil abierto, verifique las dimensiones
introducidas", MsgBoxStyle.Critical Or MsgBoxStyle.SystemModal,
"Error Critico")
Application.Exit()
            Part.Close()
        End If
```

```
        Me.Profile.Visible = False
        Me.Lines2d = Nothing
    End Sub

    Public Function CreacionPlano(Part As
SolidEdgePart.PartDocument, Perfil As SolidEdgePart.Profile,
NumPlane As Integer, Optional Dis As Double = 0) As
SolidEdgePart.Profile
        Dim ProfileSets As SolidEdgePart.ProfileSets =
Part.ProfileSets
        Dim ProfileSet As SolidEdgePart.ProfileSet =
ProfileSets.Add()
        Dim Profiles As SolidEdgePart.Profiles =
ProfileSet.Profiles
        Dim RefPlanes As SolidEdgePart.RefPlanes = Part.RefPlanes

        If Dis <> 0 Then
            Perfil =
Profiles.Add(RefPlanes.AddParallelByDistance(RefPlanes.Item(NumPla
ne), Dis, SolidEdgePart.ReferenceElementConstants.igNormalSide))
        Else
            Perfil = Profiles.Add(RefPlanes.Item(NumPlane))
        End If
        Return Perfil
    End Function

    Public Function CerrarPerfil() As
SolidEdgePart.ProfileValidationType
        Dim Relations2d As SolidEdgeFrameworkSupport.Relations2d =
Me.Profile.Relations2d
        Dim Relation2d As SolidEdgeFrameworkSupport.Relation2d =
Nothing

        Relation2d = Relations2d.AddKeypoint(Me.Lineas.Item(1),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(2),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lineas.Item(2),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(3),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lineas.Item(3),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(4),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lineas.Item(4),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(1),
KeypointIndexConstants.igLineStart)

        Return
Me.Profile.End(SolidEdgePart.ProfileValidationType.igProfileClosed
)
```

```
    End Function

    Public Sub Extrusion(Part As SolidEdgePart.PartDocument, Dis
As Double)
        Dim Models As SolidEdgePart.Models = Part.Models
        Dim Array(0 To 1) As SolidEdgePart.Profile

        Array(0) = Me.Profile
        Me.Model = Models.AddFiniteExtrudedProtrusion(1, Array,
SolidEdgePart.FeaturePropertyConstants.igRight, Dis)
    End Sub

    Public Sub PerfilCut(Part As SolidEdgePart.PartDocument,
Perfil As SolidEdgePart.Profile, ByVal D As Double)
        Me.ProfileCut = Perfil
        Me.Circles2d = Me.ProfileCut.Circles2d
        Dim Circle2d As SolidEdgeFrameworkSupport.Circle2d =
Nothing

        Circle2d = Me.Circles2d.AddByCenterRadius(0, 0, D / 2)

        If
ProfileCut.End(SolidEdgePart.ProfileValidationType.igProfileClosed
) <> 0 Then
MsgBox("Perfil abierto, verifique las dimensiones introducidas",
MsgBoxStyle.Critical Or MsgBoxStyle.SystemModal, "Error Critico")
Application.Exit()
            Part.Close()
        End If

        Me.ProfileCut.Visible = False
        Me.Circles2d = Nothing
    End Sub

    Public Sub ExtrusionCut(Part As SolidEdgePart.PartDocument,
Dis As Double)
        Dim ExtCut As SolidEdgePart.ExtrudedCutout

        ExtCut = Me.Model.ExtrudedCutouts.AddFinite(ProfileCut,
SolidEdgePart.FeaturePropertyConstants.igLeft,
SolidEdgePart.FeaturePropertyConstants.igRight, Dis)
    End Sub
End Class
```

```
Imports SolidEdgeConstants

Public Class Platina
    Protected Anc As Double
    Private Alt As Double
```

```
    Private Dia As Double
    Private Esp As Double
    Private Perfil As SolidEdgePart.Profile
    Private Perfil2 As SolidEdgePart.Profile
    Private Lineas As SolidEdgeFrameworkSupport.Lines2d
    Private Circulos As SolidEdgeFrameworkSupport.Circles2d
    Private Model As SolidEdgePart.Model
    Private Arcos As SolidEdgeFrameworkSupport.Arcs2d

    Public Sub New(ByVal Anc As Double, ByVal Alt As Double, ByVal
Dia As Double, ByVal Esp As Double)
Me.Ancho = Anc / 1000
        Me.Alto = Alt / 1000
        Me.Diametro = Dia / 1000
        Me.Espesor = Esp / 1000
End Sub

    Public Property Ancho() As Double
        Get
            Return Me.Anc
        End Get
        Set(value As Double)
            Me.Anc = value
        End Set
    End Property

    Public Property Alto() As Double
        Get
            Return Me.Alt
        End Get
        Set(value As Double)
            Me.Alt = value
        End Set
    End Property

    Public Property Diametro() As Double
        Get
            Return Me.Dia
        End Get
        Set(value As Double)
            Me.Dia = value
        End Set
    End Property

    Public Property Espesor() As Double
        Get
            Return Me.Esp
        End Get
        Set(value As Double)
            Me.Esp = value
```

```
        End Set
    End Property

    Public Property Profile() As SolidEdgePart.Profile
        Get
            Return Me.Perfil
        End Get
        Set(ByVal Value As SolidEdgePart.Profile)
            Me.Perfil = Value
        End Set
    End Property

    Public Property ProfileCut() As SolidEdgePart.Profile
        Get
            Return Me.Perfil2
        End Get
        Set(ByVal Value As SolidEdgePart.Profile)
            Me.Perfil2 = Value
        End Set
    End Property

    Public Property Lines2d() As SolidEdgeFrameworkSupport.Lines2d
        Get
            Return Me.Lineas
        End Get
        Set(ByVal Value As SolidEdgeFrameworkSupport.Lines2d)
            Me.Lineas = Value
        End Set
    End Property

    Public Property Circles2d() As
SolidEdgeFrameworkSupport.Circles2d
        Get
            Return Me.Circulos
        End Get
        Set(ByVal Value As SolidEdgeFrameworkSupport.Circles2d)
            Me.Circulos = Value
        End Set
    End Property

    Public Property Arcs2d() As SolidEdgeFrameworkSupport.Arcs2d
        Get
            Return Me.Arcos
        End Get
        Set(ByVal Value As SolidEdgeFrameworkSupport.Arcs2d)
            Me.Arcos = Value
        End Set
    End Property
```

```
    Public Sub PerfilPla(ByRef Part As SolidEdgePart.PartDocument,
ByRef Profile As SolidEdgePart.Profile)
        Me.Lines2d = Profile.Lines2d
        Dim Line2d As SolidEdgeFrameworkSupport.Line2d = Nothing
        Me.Arcs2d = Profile.Arcs2d
        Dim Arc2d As SolidEdgeFrameworkSupport.Arc2d = Nothing


        Line2d = Me.Lines2d.AddBy2Points(0, 0, Me.Ancho, 0)
        Line2d = Me.Lines2d.AddBy2Points(Me.Ancho, 0, Me.Ancho,
Me.Alto - 6 / 1000)
        Arc2d = Me.Arcs2d.AddByCenterStartEnd(Me.Ancho - 6 / 1000,
Me.Alto - 6 / 1000, Me.Ancho, Me.Alto - 6 / 1000, 0, Me.Alto - 6 /
1000)
        Line2d = Me.Lines2d.AddBy2Points(0, Me.Alto - 6 / 1000, 0,
0)

If CerrarPerfil() <> 0 Then
            MsgBox("Perfil abierto, verifique las dimensiones
introducidas", MsgBoxStyle.Critical Or MsgBoxStyle.SystemModal,
"Error Critico")
Application.Exit()
            Part.Close()
        End If

        Profile.Visible = False
        Me.Lines2d = Nothing
    End Sub

    Public Function CreacionPlano(ByRef Part As
SolidEdgePart.PartDocument, ByRef Perfil As SolidEdgePart.Profile,
NumPlane As Integer, Optional Dis As Double = 0) As
SolidEdgePart.Profile
        Dim ProfileSets As SolidEdgePart.ProfileSets =
Part.ProfileSets
        Dim ProfileSet As SolidEdgePart.ProfileSet =
ProfileSets.Add()
        Dim Profiles As SolidEdgePart.Profiles =
ProfileSet.Profiles
        Dim RefPlanes As SolidEdgePart.RefPlanes = Part.RefPlanes

        If Dis <> 0 Then
            Perfil =
Profiles.Add(RefPlanes.AddParallelByDistance(RefPlanes.Item(NumPla
ne), Dis, SolidEdgePart.ReferenceElementConstants.igNormalSide))
        Else
            Perfil = Profiles.Add(RefPlanes.Item(NumPlane))
        End If
        Return Perfil
    End Function
```

```
    Public Function CerrarPerfil() As
SolidEdgePart.ProfileValidationType
        Dim Relations2d As SolidEdgeFrameworkSupport.Relations2d =
Me.Profile.Relations2d
        Dim Relation2d As SolidEdgeFrameworkSupport.Relation2d =
Nothing

        Relation2d = Relations2d.AddKeypoint(Me.Lineas.Item(1),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(2),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lineas.Item(2),
KeypointIndexConstants.igLineEnd, Me.Arcs2d.Item(1),
KeypointIndexConstants.igArcStart)
        Relation2d = Relations2d.AddKeypoint(Me.Arcs2d.Item(1),
KeypointIndexConstants.igArcEnd, Me.Lines2d.Item(3),
KeypointIndexConstants.igLineStart)
        Relation2d = Relations2d.AddKeypoint(Me.Lineas.Item(3),
KeypointIndexConstants.igLineEnd, Me.Lines2d.Item(1),
KeypointIndexConstants.igLineStart)

        Return
Me.Profile.End(SolidEdgePart.ProfileValidationType.igProfileClosed
)
    End Function

    Public Sub Extrusion(ByRef Part As SolidEdgePart.PartDocument,
Dis As Double)
        Dim Models As SolidEdgePart.Models = Part.Models
        Dim Array(0 To 1) As SolidEdgePart.Profile

        Array(0) = Me.Profile
        Me.Model = Models.AddFiniteExtrudedProtrusion(1, Array,
SolidEdgePart.FeaturePropertyConstants.igRight, Dis)
    End Sub

    Public Sub PerfilCut(ByRef Part As SolidEdgePart.PartDocument,
ByRef Perfil As SolidEdgePart.Profile, ByVal D As Double)
        Me.Circles2d = Perfil.Circles2d
        Dim Circle2d As SolidEdgeFrameworkSupport.Circle2d =
Nothing

Circle2d = Me.Circles2d.AddByCenterRadius(Me.Ancho - 6 / 1000,
Me.Alto - 6 / 1000, D / 2)

If Perfil.End(SolidEdgePart.ProfileValidationType.igProfileClosed)
<> 0 Then
MsgBox("Perfil abierto, verifique las dimensiones introducidas",
MsgBoxStyle.Critical Or MsgBoxStyle.SystemModal, "Error Critico")
Application.Exit()
```

```
            Part.Close()
        End If


        Perfil.Visible = False
        Me.Circles2d = Nothing
    End Sub

    Public Sub ExtrusionCut(ByRef Part As
SolidEdgePart.PartDocument, Dis As Double)
        Dim ExtCut As SolidEdgePart.ExtrudedCutout

        ExtCut = Me.Model.ExtrudedCutouts.AddFinite(Me.ProfileCut,
SolidEdgePart.FeaturePropertyConstants.igLeft,
SolidEdgePart.FeaturePropertyConstants.igRight, Dis)
    End Sub

End Class
```

```
Imports System.Runtime.InteropServices
Public Class Ensamble

    Public Function CrearEnsamble() As
SolidEdgeAssembly.AssemblyDocument
        Dim objApplication As SolidEdgeFramework.Application =
Nothing
        Dim objDocuments As SolidEdgeFramework.Documents = Nothing
        Dim objAssembly As SolidEdgeAssembly.AssemblyDocument =
Nothing
        Dim objOccurrences As SolidEdgeAssembly.Occurrences =
Nothing
        Dim objOccurrence As SolidEdgeAssembly.Occurrence =
Nothing


        objApplication =
Marshal.GetActiveObject("SolidEdge.Application")
        objDocuments = objApplication.Documents
        objAssembly =
objDocuments.Add("SolidEdge.AssemblyDocument")

        Return objAssembly
    End Function

    Public Sub Ensamblar(Assembly As
SolidEdgeAssembly.AssemblyDocument, RutaImplante As String,
RutaHCor As String, RutaHTra As String, RutaPla As String, EsHCor
As Double, ByVal Lon As Double)
        Dim Occurrences As SolidEdgeAssembly.Occurrences = Nothing
```

```
        Dim Occurrence(0 To 3) As SolidEdgeAssembly.Occurrence

        Occurrences = Assembly.Occurrences

        Occurrence(0) = Occurrences.AddByFilename(RutaImplante)
        Occurrence(1) = Occurrences.AddByFilename(RutaHCor)
        Occurrence(1).Move(Lon, 0, 0)
        Occurrence(2) = Occurrences.AddByFilename(RutaHTra)
        Occurrence(2).Move(Lon + EsHCor, 0, 0)
        Occurrence(3) = Occurrences.AddByFilename(RutaPla)
        Occurrence(3).Move(1 / 1000, -6 / 1000, -13.45 / 1000)
    End Sub
End Class
```

```
Imports System.Runtime.InteropServices
Imports System.IO

Public Class Form1
    Dim objApp As SolidEdgeFramework.Application = Nothing
    Dim objDocuments As SolidEdgeFramework.Documents = Nothing
    Dim objPart As SolidEdgePart.PartDocument = Nothing

    Private Sub Bt_Dibujar_Click(sender As Object, e As EventArgs)
Handles Bt_Dibujar.Click
        Try
            If Tb_Diametro.Text <> "" And Tb_Longitud.Text <> ""
And Tb_Longitud_Roscada.Text <> "" And Tb_Paso.Text <> "" And
Tb_Nicho.Text <> "" And Tb_EHCor.Text <> "" And
(ChBox_Metrica.Checked Or (ChBox_Sierra.Checked And Tb_AI.Text <>
"")) And Tb_ASH.Text <> "" Then
                Dim RutaImplante As String = Nothing
                Dim RutaHCor As String = Nothing
                Dim RutaHTra As String = Nothing
                Dim RutaPla As String = Nothing
                BarStatus.Minimum = 0
                BarStatus.Maximum = 23
                BarStatus.Step = 1

                Dim Imp As Implante = New
Implante(Double.Parse(Tb_Diametro.Text),
Double.Parse(Tb_Longitud.Text),

Double.Parse(Tb_Longitud_Roscada.Text),
Double.Parse(Tb_Paso.Text), ChBox_Metrica.Checked,

Double.Parse(Tb_AI.Text), Double.Parse(Tb_ASH.Text))
                'Dim Imp As Implante = New Implante()
Imp.PerfilImplante(objPart, Imp.CreacionPlano(CrearParte(),
Imp.Profile, 3))
```

```
BarStatus.PerformStep()
                Imp.Revolucion(objPart, 360)
                BarStatus.PerformStep()
                Imp.PerfilHexagono(objPart,
Imp.CreacionPlano(objPart, Imp.ProfileHex, 2, 2 / 1000))
                BarStatus.PerformStep()
                Imp.ExtrusionHex(objPart, 1 / 1000)
                BarStatus.PerformStep()
                Imp.Helice(objPart, Imp.TipoRosca,
Imp.CreacionPlano(objPart, Imp.ProfileRosca, 3))
                BarStatus.PerformStep()
                Imp.Helice2(objPart, Imp.TipoRosca,
Imp.CreacionPlano(objPart, Imp.ProfileRosca, 3))
                BarStatus.PerformStep()
                Imp.PerfilFilos(objPart,
Imp.CreacionPlano(objPart, Imp.ProfileFilo, 1, -0.1 / 1000), 1)
                BarStatus.PerformStep()
                Imp.ExtrusionFilo(objPart, Imp.Diametro / 2,
SolidEdgePart.FeaturePropertyConstants.igRight)
                BarStatus.PerformStep()
                Imp.PerfilFilos(objPart,
Imp.CreacionPlano(objPart, Imp.ProfileFilo, 1, 0.1 / 1000), -1)
                BarStatus.PerformStep()
                Imp.ExtrusionFilo(objPart, Imp.Diametro / 2,
SolidEdgePart.FeaturePropertyConstants.igLeft)
                BarStatus.PerformStep()
                RutaImplante = Guardar(objPart)
                BarStatus.PerformStep()

                Dim Cor As Hueso = New
Hueso(Double.Parse(MasVariables.Tb_AlHueso.Text),
Double.Parse(MasVariables.Tb_AnHueso.Text),
Double.Parse(Tb_EHCor.Text), Double.Parse(Tb_Nicho.Text),
ChBox_Metrica.Checked, Double.Parse(Tb_ASH.Text))
                'Dim Cor As Hueso = New Hueso()
Cor.PerfilHueso(objPart, Cor.CreacionPlano(CrearParte(),
Cor.Profile, 2))
BarStatus.PerformStep()
                Cor.Extrusion(objPart, Cor.Espesor)
                BarStatus.PerformStep()
                'Cor.PerfilCut(objPart, Cor.CreacionPlano(objPart,
Cor.ProfileCut, 2), Cor.DiametroNicho)
                'BarStatus.PerformStep()
                'Cor.ExtruccionCut(objPart, Cor.Espesor)
                'BarStatus.PerformStep()
                RutaHCor = Guardar(objPart)
                BarStatus.PerformStep()

                Dim Tra As Hueso = New
Hueso(Double.Parse(MasVariables.Tb_AlHueso.Text),
```

```
Double.Parse(MasVariables.Tb_AnHueso.Text), 12,
Double.Parse(Tb_Nicho.Text), ChBox_Metrica.Checked,
Double.Parse(Tb_ASH.Text))
'Dim Tra As Hueso = New Hueso()
                Tra.PerfilHueso(objPart,
Tra.CreacionPlano(CrearParte(), Tra.Profile, 2))
BarStatus.PerformStep()
                Tra.Extrusion(objPart, Tra.Espesor)
                BarStatus.PerformStep()
                'Tra.PerfilCut(objPart, Tra.CreacionPlano(objPart,
Tra.ProfileCut, 2), Tra.DiametroNicho)
                'BarStatus.PerformStep()
                'Tra.ExtruccionCut(objPart, Tra.Espesor)
                'BarStatus.PerformStep()
                RutaHTra = Guardar(objPart)
                BarStatus.PerformStep()

Dim Pla As Platina = New Platina(12, 18, 4.3, 1)
                Pla.PerfilPla(objPart,
Pla.CreacionPlano(CrearParte(), Pla.Profile, 2))
BarStatus.PerformStep()
                Pla.Extruccion(objPart, Pla.Espesor)
                BarStatus.PerformStep()
                Pla.PerfilCut(objPart, Pla.CreacionPlano(objPart,
Pla.ProfileCut, 2), Pla.Diametro)
                BarStatus.PerformStep()
                Pla.ExtruccionCut(objPart, Pla.Espesor)
                BarStatus.PerformStep()
                RutaPla = Guardar(objPart)
                BarStatus.PerformStep()

                Dim Ens As Ensamble = New Ensamble()
Ens.Ensamblar(Ens.CrearEnsamble(), RutaImplante, RutaHCor,
RutaHTra, RutaPla, Cor.Espesor, (3 / 1000) + (Imp.Longitud -
Imp.LongRos))
BarStatus.PerformStep()

                Application.Exit()
            Else
                MsgBox("Todos los datos son obligatorios")
            End If
        Catch ex As Exception
            MsgBox(ex.ToString)
            Application.Exit()
            objPart.Close()
        End Try
    End Sub

    'Funciones Control
```

```
    Private Sub Form1_KeyPress(sender As Object, e As
KeyPressEventArgs) Handles Me.KeyPress
        If e.KeyChar = ChrW(Keys.Escape) Then
            Tb_Diametro.Text = Nothing
            Tb_Longitud.Text = Nothing
            Tb_Longitud_Roscada.Text = Nothing
            Tb_Paso.Text = Nothing
            Me.Close()
            Application.Exit()
        End If
    End Sub

    Private Sub ChBox_Metrica_CheckedChanged(sender As Object, e
As EventArgs) Handles ChBox_Metrica.CheckedChanged
        If ChBox_Metrica.Checked = True Then
            ChBox_Sierra.Enabled = False
        ElseIf ChBox_Metrica.Checked = False Then
            ChBox_Sierra.Enabled = True
        End If
    End Sub

    Private Sub ChBox_Sierra_CheckedChanged(sender As Object, e As
EventArgs) Handles ChBox_Sierra.CheckedChanged
        If ChBox_Sierra.Checked = True Then
            ChBox_Metrica.Enabled = False
            Tb_AI.Enabled = True
        ElseIf ChBox_Sierra.Checked = False Then
            ChBox_Metrica.Enabled = True
            Tb_AI.Text = Nothing
            Tb_AI.Enabled = False
        End If
    End Sub

    Private Sub Tb_Diametro_LostFocus(sender As Object, e As
EventArgs) Handles Tb_Diametro.LostFocus
        Try
            If Tb_Diametro.Text <> "" Then
                If Tb_Diametro.Text.Contains(".") = True Then
                    Tb_Diametro.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
Tb_Diametro.Focus()
                Else
                    If Double.Parse(Tb_Diametro.Text) < 1 Or
Double.Parse(Tb_Diametro.Text) > 2 Then
Tb_Diametro.BackColor = Color.Red
                        MsgBox("El diametro esta fuera del rango
(1 mm - 2 mm)", MsgBoxStyle.Exclamation Or
MsgBoxStyle.SystemModal, "Alerta")
Tb_Diametro.Focus()
```

```
                    Else
                        Tb_Diametro.BackColor = Color.White
                    End If
                End If
            End If
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try
    End Sub

    Private Sub Tb_Longitud_LostFocus(sender As Object, e As
EventArgs) Handles Tb_Longitud.LostFocus
        Try
            If Tb_Longitud.Text <> "" Then
                If Tb_Longitud.Text.Contains(".") = True Then
                    Tb_Longitud.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
Tb_Longitud.Focus()
                Else
                    If Double.Parse(Tb_Longitud.Text) < 5 Or
Double.Parse(Tb_Longitud.Text) > 12 Then
                        'Tb_Longitud.BackColor = Color.Red
                        'MsgBox("La longitud esta fuera del rango
(5 mm - 12 mm)", MsgBoxStyle.Exclamation Or
MsgBoxStyle.SystemModal, "Alerta")
                        'Tb_Longitud.Focus()
                    Else
                        Tb_Longitud.BackColor = Color.White
                    End If
                End If
            End If
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try
    End Sub

    Private Sub Tb_Longitud_Roscada_LostFocus(sender As Object, e
As EventArgs) Handles Tb_Longitud_Roscada.LostFocus
        Try
            If Tb_Longitud_Roscada.Text <> "" Then
                If Tb_Longitud_Roscada.Text.Contains(".") = True
Then
                    Tb_Longitud_Roscada.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
                    Tb_Longitud_Roscada.Focus()
                Else
                    'MsgBox(Tb_Longitud.Text, ",",
Tb_Longitud_Roscada.Text)
```

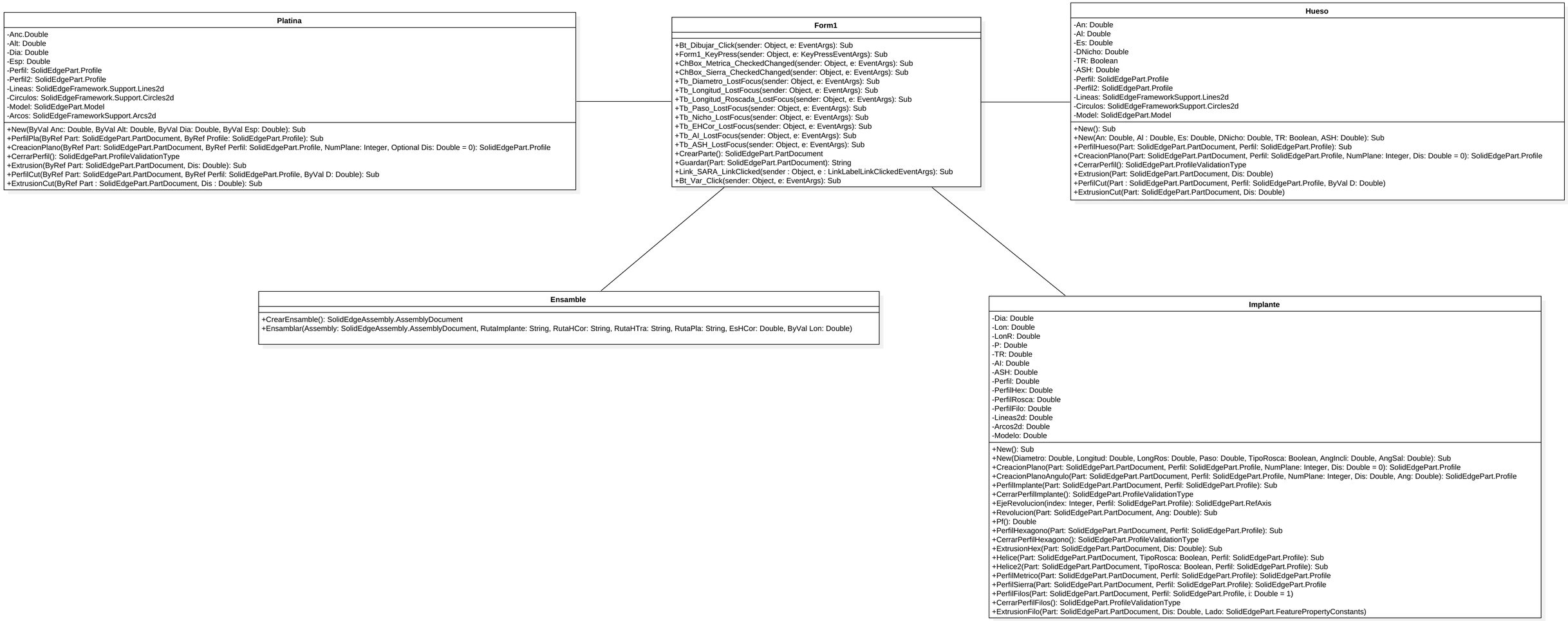
```
If Double.Parse(Tb_Longitud.Text) <=
Double.Parse(Tb_Longitud_Roscada.Text) Then
Tb_Longitud_Roscada.BackColor = Color.Red
                        MsgBox("La longitud roscada no puede ser
mayor a la longitud", MsgBoxStyle.Exclamation Or
MsgBoxStyle.SystemModal, "Alerta")
Tb_Longitud_Roscada.Focus()
                    Else
                        Tb_Longitud_Roscada.BackColor =
Color.White
                    End If
                End If
            End If
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try
    End Sub

    Private Sub Tb_Paso_LostFocus(sender As Object, e As
EventArgs) Handles Tb_Paso.LostFocus
        If Tb_Paso.Text <> "" Then
            If Tb_Paso.Text.Contains(".") = True Then
                Tb_Paso.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
Tb_Paso.Focus()
            Else
                If Double.Parse(Tb_Paso.Text) <= 0 Or
Double.Parse(Tb_Paso.Text) > 1 Then
Tb_Paso.BackColor = Color.Red
                    MsgBox("El paso esta fuera del rango (0.1 mm -
1 mm)", MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal,
"Alerta")
Tb_Paso.Focus()
                Else
                    Tb_Paso.BackColor = Color.White
                End If
            End If
        End If
    End Sub

    Private Sub Tb_Nicho_LostFocus(sender As Object, e As
EventArgs) Handles Tb_Nicho.LostFocus
        If Tb_Nicho.Text <> "" Then
            If Tb_Nicho.Text.Contains(".") = True Then
                Tb_Nicho.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
Tb_Nicho.Focus()
            Else
```

```
                If Double.Parse(Tb_Nicho.Text) <= 0 Or
Double.Parse(Tb_Nicho.Text) >= Double.Parse(Tb_Diametro.Text) Then
Tb_Nicho.BackColor = Color.Red
                    MsgBox("El paso esta fuera del rango (0.1 mm -
Diametro)", MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal,
"Alerta")
Tb_Nicho.Focus()
                Else
                    Tb_Nicho.BackColor = Color.White
                End If
            End If
        End If
    End Sub

    Private Sub Tb_EHCor_LostFocus(sender As Object, e As
EventArgs) Handles Tb_EHCor.LostFocus
        If Tb_EHCor.Text <> "" Then
            If Tb_EHCor.Text.Contains(".") = True Then
                Tb_EHCor.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
Tb_EHCor.Focus()
            Else
                If Double.Parse(Tb_EHCor.Text) <= 0 Or
Double.Parse(Tb_EHCor.Text) > 5 Then
                    Tb_EHCor.BackColor = Color.Red
                    MsgBox("El paso esta fuera del rango (0.1 mm -
5 mm)", MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal,
"Alerta")
                    Tb_EHCor.Focus()
                Else
                    Tb_EHCor.BackColor = Color.White
                End If
            End If
        End If
    End Sub

    Private Sub Tb_AI_LostFocus(sender As Object, e As EventArgs)
Handles Tb_AI.LostFocus
        If Tb_AI.Text <> "" Then
            If Tb_Paso.Text.Contains(".") = True Then
                Tb_Paso.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
Tb_Paso.Focus()
            Else
                If Double.Parse(Tb_AI.Text) < 5 Or
Double.Parse(Tb_AI.Text) >= 30 Then
Tb_Paso.BackColor = Color.Red
```

```
                    MsgBox("El angulo de salida esta fuera del
rango (5° - 30°)", MsgBoxStyle.Exclamation Or
MsgBoxStyle.SystemModal, "Alerta")
Tb_Paso.Focus()
                End Else
                    Tb_Paso.BackColor = Color.White
                End If
            End If
        End If
    End Sub

    Private Sub Tb_ASH_LostFocus(sender As Object, e As EventArgs)
Handles Tb_ASH.LostFocus
        If Tb_ASH.Text <> "" Then
            If Tb_Paso.Text.Contains(".") = True Then
                Tb_Paso.BackColor = Color.Red
MsgBox("No se permiten puntos como separador decimal",
MsgBoxStyle.Exclamation Or MsgBoxStyle.SystemModal, "Alerta")
Tb_Paso.Focus()
            Else
                If Double.Parse(Tb_ASH.Text) < 5 Or
Double.Parse(Tb_ASH.Text) > 30 Then
Tb_Paso.BackColor = Color.Red
                    MsgBox("El angulo de salida esta fuera del
rango (5° - 30°)", MsgBoxStyle.Exclamation Or
MsgBoxStyle.SystemModal, "Alerta")
Tb_Paso.Focus()
                Else
                    Tb_Paso.BackColor = Color.White
                End If
            End If
        End If
    End Sub

    'Private Sub Form1_Load(sender As Object, e As EventArgs)
Handles MyBase.Load
    '    Me.TopMost = True
    'End Sub

'Funciones con logica

    Public Function CrearParte() As SolidEdgePart.PartDocument
objApp = Marshal.GetActiveObject("SolidEdge.Application")
        objDocuments = objApp.Documents
        objPart = objDocuments.Add("SolidEdge.PartDocument")
        objApp.DisplayAlerts = True
        Return objPart
    End Function
```

```
    Public Function Guardar(Part As SolidEdgePart.PartDocument) As
String
        Dim Path As String = Nothing
        Try
            Part.Save()
            Path = Part.FullName.ToString
            Part.Close()
            Return Path
        Catch ex As Exception
            If ex.ToString.Contains("0x80004004") Then
                Guardar(Part)
            Else
                MsgBox(ex.ToString, MsgBoxStyle.Critical Or
MsgBoxStyle.SystemModal, "Error Critico")
            End If
        End Try
    End Function

    Private Sub Link_SARA_LinkClicked(sender As Object, e As
LinkLabelLinkClickedEventArgs) Handles Link_SARA.LinkClicked
        Try
            Link_SARA.LinkVisited = True

System.Diagnostics.Process.Start("http://semillerosara.wixsite.com
/sarasemillero")
        Catch ex As Exception
            MsgBox(ex.ToString, MsgBoxStyle.Critical Or
MsgBoxStyle.SystemModal, "Error Critico")
        End Try
    End Sub

    Private Sub Bt_Var_Click(sender As Object, e As EventArgs)
Handles Bt_Var.Click
        MasVariables.Show()
    End Sub

End Class
```

B. Anexo:Diagrama de clases


Platina
-Anc.Double
-Alt: Double
-Dia: Double
-Esp: Double
-Perfil: SolidEdgePart.Profile
-Perfil2: SolidEdgePart.Profile
-Lineas: SolidEdgeFramework.Support.Lines2d
-Circulos: SolidEdgeFramework.Support.Circles2d
-Model: SolidEdgePart.Model
-Arcos: SolidEdgeFrameworkSupport.Arcs2d
+New(ByVal Anc: Double, ByVal Alt: Double, ByVal Dia: Double, ByVal Esp: Double): Sub
+PerfilPla(ByRef Part: SolidEdgePart.PartDocument, ByRef Profile: SolidEdgePart.Profile): Sub
+CreacionPlano(ByRef Part: SolidEdgePart.PartDocument, ByRef Perfil: SolidEdgePart.Profile, NumPlane: Integer, Optional Dis: Double = 0): SolidEdgePart.Profile
+CerrarPerfil(): SolidEdgePart.ProfileValidationType
+Extrusion(ByRef Part: SolidEdgePart.PartDocument, Dis: Double): Sub
+PerfilCut(ByRef Part: SolidEdgePart.PartDocument, ByRef Perfil: SolidEdgePart.Profile, ByVal D: Double): Sub
+ExtrusionCut(ByRef Part : SolidEdgePart.PartDocument, Dis : Double): Sub
Form1
+Bt_Dibujar_Click(sender: Object, e: EventArgs): Sub
+Form1_KeyPress(sender: Object, e: KeyPressEventArgs): Sub
+ChBox_Metrica_CheckedChanged(sender: Object, e: EventArgs): Sub
+ChBox_Sierra_CheckedChanged(sender: Object, e: EventArgs): Sub
+Tb_Diametro_LostFocus(sender: Object, e: EventArgs): Sub
+Tb_Longitud_LostFocus(sender: Object, e: EventArgs): Sub
+Tb_Longitud_Roscada_LostFocus(sender: Object, e: EventArgs): Sub
+Tb_Paso_LostFocus(sender: Object, e: EventArgs): Sub
+Tb_Nicho_LostFocus(sender: Object, e: EventArgs): Sub
+Tb_EHCor_LostFocus(sender: Object, e: EventArgs): Sub
+Tb_Al_LostFocus(sender: Object, e: EventArgs): Sub
+Tb_ASH_LostFocus(sender: Object, e: EventArgs): Sub
+CrearParte(): SolidEdgePart.PartDocument
+Guardar(Part: SolidEdgePart.PartDocument): String
+Link_SARA_LinkClicked(sender : Object, e : LinkLabelLinkClickedEventArgs): Sub
+Bt_Var_Click(sender: Object, e: EventArgs): Sub
Hueso
-An: Double
-Al: Double
-Es: Double
-DNicho: Double
-TR: Boolean
-ASH: Double
-Perfil: SolidEdgePart.Profile
-Perfil2: SolidEdgePart.Profile
-Lineas: SolidEdgeFrameworkSupport.Lines2d
-Circulos: SolidEdgeFrameworkSupport.Circles2d
-Model: SolidEdgePart.Model
+New(): Sub
+New(An: Double, Al : Double, Es: Double, DNicho: Double, TR: Boolean, ASH: Double): Sub
+PerfilHueso(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile): Sub
+CreacionPlano(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile, NumPlane: Integer, Dis: Double = 0): SolidEdgePart.Profile
+CerrarPerfil(): SolidEdgePart.ProfileValidationType
+Extrusion(Part: SolidEdgePart.PartDocument, Dis: Double)
+PerfilCut(Part : SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile, ByVal D: Double)
+ExtrusionCut(Part: SolidEdgePart.PartDocument, Dis: Double)
Ensamble
+CrearEnsamble(): SolidEdgeAssembly.AssemblyDocument
+Ensamblar(Assembly: SolidEdgeAssembly.AssemblyDocument, RutaImplante: String, RutaHCor: String, RutaHTra: String, RutaPla: String, EsHCor: Double, ByVal Lon: Double)
Implante
-Dia: Double
-Lon: Double
-LonR: Double
-P: Double
-TR: Double
-Al: Double
-ASH: Double
-Perfil: Double
-PerfilHex: Double
-PerfilRosca: Double
-PerfilFilo: Double
-Lineas2d: Double
-Arcos2d: Double
-Modelo: Double
+New(): Sub
+New(Diametro: Double, Longitud: Double, LongRos: Double, Paso: Double, TipoRosca: Boolean, AngIncli: Double, AngSal: Double): Sub
+CreacionPlano(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile, NumPlane: Integer, Dis: Double = 0): SolidEdgePart.Profile
+CreacionPlanoAngulo(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile, NumPlane: Integer, Dis: Double, Ang: Double): SolidEdgePart.Profile
+PerfilImplante(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile): Sub
+CerrarPerfilImplante(): SolidEdgePart.ProfileValidationType
+EjeRevolucion(index: Integer, Perfil: SolidEdgePart.Profile): SolidEdgePart.RefAxis
+Revolucion(Part: SolidEdgePart.PartDocument, Ang: Double): Sub
+Pf(): Double
+PerfilHexagono(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile): Sub
+CerrarPerfilHexagono(): SolidEdgePart.ProfileValidationType
+ExtrusionHex(Part: SolidEdgePart.PartDocument, Dis: Double): Sub
+Helice(Part: SolidEdgePart.PartDocument, TipoRosca: Boolean, Perfil: SolidEdgePart.Profile): Sub
+Helice2(Part: SolidEdgePart.PartDocument, TipoRosca: Boolean, Perfil: SolidEdgePart.Profile): Sub
+PerfilMetrico(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile): SolidEdgePart.Profile
+PerfilSierra(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile): SolidEdgePart.Profile
+PerfilFilos(Part: SolidEdgePart.PartDocument, Perfil: SolidEdgePart.Profile, i: Double = 1)
+CerrarPerfilFilos(): SolidEdgePart.ProfileValidationType
+ExtrusionFilo(Part: SolidEdgePart.PartDocument, Dis: Double, Lado: SolidEdgePart.FeaturePropertyConstants)

C. Anexo:Paper CIMM 2017

VIII Congreso Internacional de Ingenierías Mecánica y Mecatrónica y VI de Materiales, Energía y Medio Ambiente
26, 27 y 28 de abril de 2017 – Medellín, Colombia

IMPLEMENTACIÓN DEL PROCESO DE MANUFACTURA FLEXIBLE EN LA FABRICACIÓN DE UN MINI-IMPLANTE PARA ORTODONCIA

Juan David Ramírez Vásquez (1), John Alexander Zambrano Pulido (1), Oscar Rodrigo López Vaca (1), Jorge Andrés García Barbosa (1)

1. Universidad Santo Tomás, Colombia

Núcleo Temático: Materiales y Procesos.

Resumen

Se realizó la manufactura de un mini-implante para aplicaciones en ortodoncia, de diferentes dimensiones, personalizado de acuerdo a las características del paciente, en un proceso de producción continuo y flexible, utilizando herramientas de programación paramétrica. El cambio de parámetros en el control del torno tipo suizo se logró por medio de una aplicación externa desarrollada en Visual Basic, que generó un programa de producción en función de los requerimientos de cada cliente.

Introducción

El aumento en la competitividad de los mercados, en la calidad de los bienes fabricados y en la demanda de productos personalizados, enfocan a una mayor implementación de manufactura flexible en las plantas de producción [Brettel *et al*, 2016] [Kusiak, 2000]. En el área de la ortodoncia, a consecuencia de la variabilidad de la densidad ósea del maxilar superior y del maxilar inferior, además del espesor de las capas óseas (cortical y trabecular), se generan variaciones en los requerimientos geométricos funcionales en la fabricación de mini-implantes, tales como diámetro, longitud, paso y perfil de la rosca [Kim *et al*, 2014]. En la producción de componentes cuyas características geométricas son variables, las empresas no alcanzan a cubrir una gran variedad y por el contrario disponen en el mercado de un número reducido de tipos de componentes [Cox *et al*, 1989]. Por tal motivo se hace indispensable explorar procesos de manufactura que permitan la fabricación de productos personalizados pero con características de producción en serie.

Métodos

El proceso de manufactura flexible se implementó en un torno suizo de control numérico marca Polygim (Ref. Diamond 20CSB), gobernado por un control Mitsubishi (Ref. Meldas 500M), con capacidades de programación paramétrica. Se utilizaron herramientas especializadas para el mecanizado de pequeños componentes en la aleación de titanio Ti-6Al-4V.

El aplicativo que integra los requerimientos personalizados de cada cliente, para la generación del programa de producción en código G, se desarrolló en Visual Basic. Dicho programa de producción se envió al controlador de la máquina, desde un computador, utilizando un programa de transmisión de datos estándar.

De acuerdo al diseño del producto a manufacturar (Figura 1), se programó en código G la fabricación de una de las posibles configuraciones de mini-implante en cuanto a diámetro, a longitud, a longitud no roscada y a paso. Debido a que las variaciones en los requerimientos geométricos funcionales son en valor y no en forma, además de que la secuencia de producción para su fabricación no varía, las múltiples configuraciones de un mini-implante conforman una familia de partes [Djassemi, 1998]. Lo anterior permite abordar parte de la problemática con la parametrización del código G. Con este código inicial es posible hacer un panorama de las variables a utilizar y de la posible creación de variables auxiliares para cálculos necesarios tales como profundidad del filete de rosca o la longitud roscada.

En la Figura 1 se observa en detalle las variables correspondientes a las características geométricas del mini-implante requerido. Las variables a parametrizar son el diámetro, la longitud, la longitud no roscada y el paso. A cada una de estas características geométricas se le asigna una variable en el controlador.

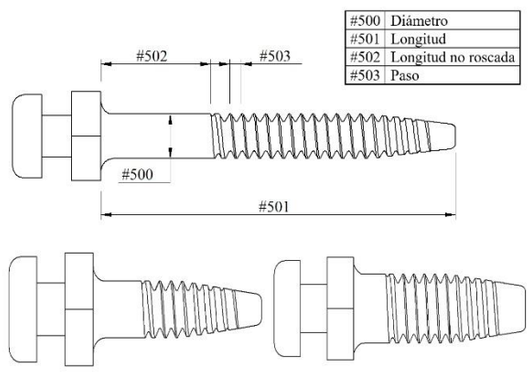


Figura 1: Parámetros geométricos del mini implante y ejemplos de la familia de partes

Como se mencionó anteriormente, la parametrización del código solo aborda parte de la solución, debido a que agrega flexibilidad a la producción [García *et al*, 2014], pero no continuidad. Esto como consecuencia de que después de la finalización de la producción de una pieza, se requiere de la modificación manual de las variables en el controlador de la máquina para la manufactura de la siguiente, que tiene características geométricas diferentes.

Para lograr un proceso de producción continuo se hizo uso de la técnica de programación con subprogramas. Por lo que se desarrollaron dos códigos; El primero, denominado código residente, contenía la parametrización del código inicial, además del cálculo de las variables auxiliares. El código residente se almacenó en la memoria del controlador y permaneció invariable durante la ejecución del proceso de manufactura de la familia de partes; El segundo código, se denominó código del programa de producción y contenía la modificación a las variables principales en función de los

requerimientos de cada cliente, junto con el llamado sucesivo del código residente a través de una función especializada del controlador (M98).
La elaboración del código G del programa de producción se realizó con ayuda de un aplicativo computacional desarrollado en Visual Basic. En el aplicativo se hace recopilación de los requerimientos de cada cliente a través de un control de muestreo de datos en formato tabla (*Data View Grid*) como se muestra en la Figura 2.

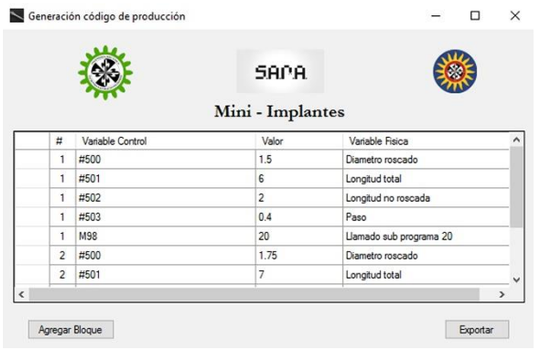


Figura 2. Aplicativo desarrollado en Visual Basic

Recopilados los requerimientos de suficientes clientes para la fabricación de un lote de piezas, el aplicativo genera un archivo en código G, que es enviado a través del puerto de comunicación RS232 al controlador de la máquina, por medio de un programa de transmisión de datos estándar. El código del programa de producción es sustituido por uno nuevo, una vez finalizado el proceso de manufactura.

Resultados

La Figura 3. presenta la metodología de implementación de un proceso de manufactura flexible y continuo para la obtención de lotes de familias de partes con diferentes características geométricas.

La Figura 4. muestra los resultados de producción del proceso de manufactura flexible y continúo implementado. Los mini-implantes presentados tienen diferentes configuraciones de características geométricas y fueron fabricados en la aleación de titanio Ti-6Al-4V bajo las condiciones de corte sugeridas por el fabricante de las herramientas.

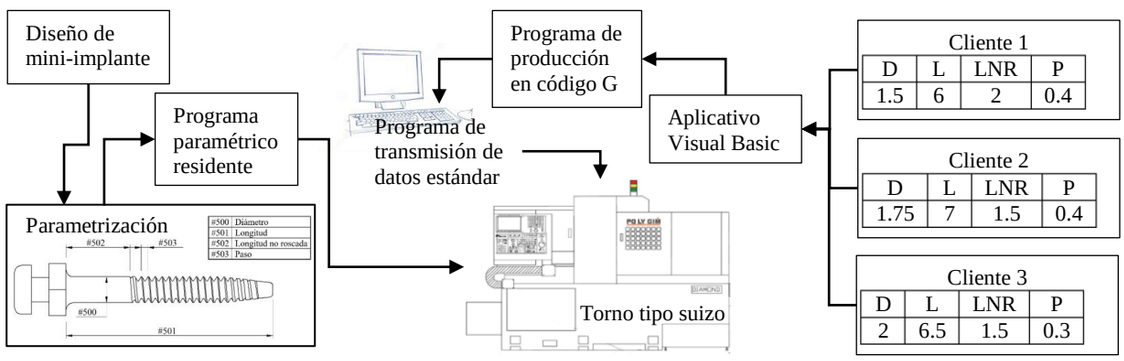


Figura 3. Metodología de implementación de un proceso de manufactura flexible y continuo. D: Diámetro, L:Longitud, LNR: Longitud no roscada, P: Paso.

El tiempo de producción en promedio, por mini-implante, fue de 1 m 27 s sin incluir el tiempo en que el operario o el código de producción configura las variables y llama de nuevo al código residente. Es de tener en cuenta que si el cambio de parámetros se realiza de forma manual el tiempo en adición por esta operación es de 45 s, en caso contrario el tiempo en adición es prácticamente nulo.

Figura 4. Familia de partes producidas con diferentes características geométricas

Conclusiones

La programación paramétrica embebida en los controladores CNC, junto con aplicativos desarrollados en plataformas de programación, dieron como resultado un proceso de fabricación ágil y flexible, de familias de mini-implantes en función de sus principales características geométricas.

Los tiempos de producción fueron reducidos en un 34.1% al eliminar el cambio de parámetros de forma manual, el cual era equivalente al 51.7% del tiempo de maquinado.

Agradecimiento

Los autores agradecen el financiamiento recibido por parte de la Universidad Santo Tomás a través del Fondo para la Investigación FODEIN, bajo códigos de proyecto 1605501 – 019 y 1605502 – 020.

Referencias

Brettel *et al*, Procedia CIRP, 41:105–110, 2016.

Cox *et al*, Production and Inventory Management Journal, 68-72, 1989.

Djassemi M, A parametric programming technique for efficient CNC machining operations, Computational Industrial Engineering, 35:33-36, 1998.

García-Barbosa J. A, Simulation and Verification of parametric numerical control programs using a virtual machine tool, Production Engineering, 8:407-413, 2014

Kim *et al*, Temporary skeletal Anchorage Devices, Ed. Springer-Verlag Berlin Heidelberg, 2014.

Kusiak Andrew, Computational Intelligence in Design and Manufacturing, John Wiley and Sons, United States of America, 2000.

D. Anexo:Paper CIMM 2017

VIII Congreso Internacional de Ingenierías Mecánica y Mecatrónica y VI de Materiales, Energía y Medio Ambiente
26, 27 y 28 de abril de 2017 – Medellín, Colombia

DETERMINACIÓN DE LA ESTABILIDAD PRIMARIA EN MINI-IMPLANTES EN FUNCIÓN DE SU GEOMETRÍA Y LA CALIDAD DEL HUESO MAXILAR.

John Zambrano (1), Rodrigo López (2), Juan Ramírez (3), Jorge García (4) Liliana Jara (5)

1-4. Universidad Santo Tomás, Colombia
5- Institución universitaria colegios de Colombia - UNICOC

Área Temática: Biomecánica

Resumen

El propósito de éste trabajo fue evaluar mediante herramientas computacionales la rigidez de la interface hueso-MI, para ello se desarrollaron 16 configuraciones distintas en donde se varía la geometría del mini implante (paso de rosca, diámetro y longitud roscada) y el espesor de hueso cortical. Se dio como resultado el valor de los esfuerzos máximos en la zona cortical y trabecular para cada una de las configuraciones, así como también la distribución de esfuerzos y el cambio en los valores de esfuerzo a lo largo del espesor de hueso cortical y la longitud de hueso trabecular. El estudio demostró que entre mayor sea el paso de rosca los esfuerzos máximos aumentan y la distribución de esfuerzos se presenta no homogénea en la interfaz. Se evidenció que el espesor de hueso cortical influye siempre y cuando el paso de rosca sea corto y que la longitud roscada incide más en la distribución de esfuerzos de la zona trabecular que en una disminución de los valores de esfuerzo máximos de la zona cortical.

Introducción

Los mini-implantes (MI) son pequeños dispositivos manufacturados en aleación de titanio (Ti-6Al-4V) grado 23 o acero quirúrgico que se implantan generalmente sobre el hueso basal y logran un anclaje absoluto en las mecánicas ortodónticas. Fueron usados por primera vez hace 70 años en un experimento para retraer los caninos [Gainsforth, 1945], en él se demostró que los movimientos dentales por medio de un anclaje en el hueso basal eran posibles. Los MI no fueron bien aceptados ya que generaban costos mayores por implantación y extracción, tiempos de espera a carga mayores y un estigma del paciente a ser más doloroso e incómodo que otros accesorios [Buschang, 2008]. Es por ello que los MI han sido continuamente modificados con el fin de ajustarse a las tendencias clínicas. Actualmente, su tamaño los hace tan versátiles que pueden ser ubicados en múltiples regiones, pueden ser insertados y removidos rápidamente sin causar molestias y se pueden cargar inmediatamente al anclarse [Reynders, 2009].

Sin embargo, existen algunos inconvenientes con el uso de los MI en lo que la literatura denomina estabilidad primaria, la cual se define como la obstrucción mecánica del MI en el hueso obtenida al instante de la inserción [Martínez-Gonzalez, 2002]. La estabilidad primaria es de vital importancia para los procedimientos ortodónticos a largo plazo ya que asegura el tiempo en el cual la oseointegración se hace presente en la interfaz hueso- MI y así lograr un anclaje absoluto. Se ha reportado un porcentaje de fracaso entre el 10% y el 30% [Cunha, 2015], y este es atribuido a factores biológicos (gingivitis e inflamación del tejido periodontal) y mecánicos (estabilidad primaria y preparación de cavidad), lo cual provee incentivo a determinar los factores necesarios que optimicen el uso de MI ortodónticos. Por lo anterior, con el objetivo de determinar la variación en la estabilidad primaria de los MI, en este trabajo se realizó, mediante el uso de herramientas computacionales, el estudio de la afectación de la estabilidad primaria variando las propiedades geométricas del cuerpo roscado del MI (longitud, paso de la rosca y diámetro), teniendo en cuenta la calidad ósea del maxilar en donde son generalmente implantados.

Métodos

Para el estudio del comportamiento de la interface MI-hueso, inicialmente se desarrolló un ensayo experimental con 16 MIs comerciales (diámetro 1.8mm y longitud 8mm) con iguales características geométricas, implantados en costillas de cerdo (en zonas biológicamente similares determinadas mediante tomografías computacionales (TC)), los cuales fueron sometidos a cargas de flexión en una máquina de ensayos universales como se muestra en la [Figura 1a]. Como resultado se obtuvo la resistencia máxima del conjunto MI-hueso y el comportamiento de las deformaciones producidas por el MI sobre el hueso. Estos resultados experimentales fueron el insumo para la realización y validación de la experimentación in-silico, utilizando un modelo de elementos finitos (EF) 3-dimensional, del MI insertado en una sección de hueso maxilar en donde se diferenció el hueso trabecular y el cortical [Figura 1b].

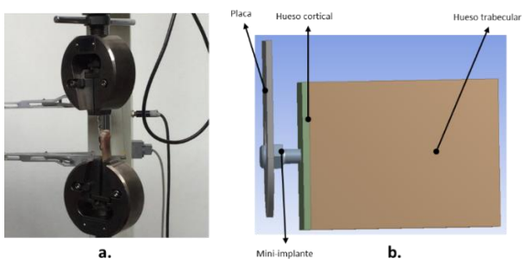
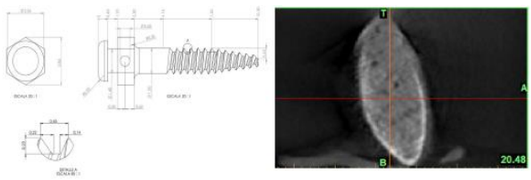


Figura 1. a. montaje MI-hueso en máquina de ensayos universales. b. Modelo CAD 3-dimensional MI-hueso.

Para un correcto modelamiento del dominio inicialmente se realizó la metrología de los implantes, utilizando para ello un proyector de perfiles horizontal (MITUTOYO PH-3515 F), la metrología permitió realizar el plano del MI que se observa en la [Figura 2a] y luego se determinó el rango de espesor de hueso cortical mediante la visualización de los valores arrojados en las TC (0.8 a 1,7 mm) [Figura 2b].



a. b.

Figura 2. a. Plano del dispositivo de anclaje temporal.b. Tomografía de costilla de cerdo

El tejido óseo (trabecular y cortical) presenta un comportamiento visco-poro-elástico, permitiéndole lograr una deformación excesiva antes de llegar a plasticidad, sin embargo éste comportamiento mecánico es complejo dado que necesita una caracterización numérica de la relajación de carga y la deformación por fluencia (creep), es por ello que dada la complejidad de este modelo de material se plantea realizar una representación mecánica más simple para el hueso. Por lo anterior, se decide tomar un modelo bilineal que permita ingresar los valores elásticos y plásticos del hueso (Trabecular y cortical) con el fin de lograr una simulación computacional aproximada al comportamiento del hueso.

Parte	Módulo elástico E (MPa)	Relación de Poisson	Límite Elástico (MPa)	Módulo Tangente (MPa)
Hueso Cortical	12100	0,26	133	80
Hueso Trabecular	282	0,3	1,902	10,88
Placa (Acero Inoxidable)	193000	0,31	207	-
Mini-implante (Titanio)	96000	0,36	930	-

Tabla 1. Condiciones de material utilizados en el modelo computacional [Kong , 2006] [Ammar, 2011]. Nota: los valores utilizados para el hueso trabecular se determinaron mediante una gráfica fuerza-desplazamiento determinada en una prueba experimental.

Los modelos computacionales fueron discretizados utilizando elementos tridimensionales tetraédricos, se hizo énfasis en la zona roscada del hueso (Trabecular-cortical) junto con la geometría del implante. Se definió un tamaño de elemento para la zona roscada del hueso de 0,2mm y para el mini implante 0,5mm. El resultado final fue un conjunto de modelos con un rango de 33.000 – 49.000 nodos y 18.800 – 29.000 elementos. El número de nodos y elementos utilizado para cada modelo se describe en la [Tabla 2].

Modelo	# Nodos	#Elementos
1,4 - 8 - 0, 5 - 1	37677	21674
1,4 - 8 - 0,75 - 1	35372	20357
1,4 - 10 - 0,5 - 1	45751	26662
1,4 - 10 - 0,75 - 1	37613	21868
1,5 - 8 - 0,5 - 1	39523	22833
1,5 - 8 - 0,75 - 1	33050	18988
1,5 - 10 - 0,5 - 1	48325	28215
1,5 - 10 - 0,75 - 1	40132	23354
1,4 - 8 - 0,5 - 1,5	39173	22660
1,4 - 8 - 0,75 - 1,5	35891	20685
1,4 - 10 - 0,5 - 1,5	46464	27138
1,4 - 10 - 0,75 - 1,5	38024	22123
1,5 - 8 - 0,5 - 1,5	40052	23117
1,5 - 8 - 0,75 - 1,5	33504	19249
1,5 - 10 - 0,5 - 1,5	48869	28539
1,5 - 10 - 0,75 - 1,5	40659	23693

Tabla 2. Número total de nodos y elementos utilizados para cada modelo.

Como condiciones de contacto se establecieron los comportamientos mecánicos de cada interface acorde a los tratamientos ortodónticos usados. Los tipos de contacto utilizados se presentan en la [Tabla 3]. Como condiciones de frontera se aplicó una presión de 5 MPa sobre la cara plana superior de la placa que la tensiona y se tomaron como soportes fijos las caras planas exteriores de la geometría usada como hueso cortical y trabecular.

Definición de contacto	
Interface	***Tipo de contacto***
Placa - Mini implante	Bonded
H.Cortical - Mini implante	Fricción (μ = 0,37)
H.Trabecular - Mini implante	Fricción (μ = 0,37)
H.Cortical - H.Trabecular	Bonded

Tabla 3. Contactos usados para las distintas interfaces del modelo. Los valores del coeficiente de fricción fueron tomados de [Ammar, 2011].

Finalmente se obtuvo un conjunto de 16 modelos computacionales entre los cuales se varió el diámetro (1,4 – 1,5 mm), la longitud roscada (8 – 10 mm), el paso (0,5 – 0,75 mm) y el espesor de hueso cortical (1 – 1,5 mm).

Resultados

Se dio como resultado la distribución de esfuerzos y deformaciones para cada una de las configuraciones geométricas establecidas, gracias a esto se puede definir cuál de todas las variaciones dispuestas es la que afecta en mayor o menor medida el valor de los comportamientos mecánicos en la zona de la interfaz. Los valores máximos de esfuerzo de von-Mises para el hueso cortical y trabecular se muestran en la [Tabla 4].

		Esfuerzo máximo (MPa)	
#	***Modelo***	***Cortical***	***Trabecular***
1	1,4 - 8 - 0, 5 - 1	143,95	3,084
2	1,4 - 8 - 0,75 - 1	202,58	3,282
3	1,4 - 10 - 0,5 - 1	147,55	2,65
4	1,4 - 10 - 0,75 - 1	161,4	2,7358
5	1,5 - 8 - 0,5 - 1	146,22	2,5212
6	1,5 - 8 - 0,75 - 1	176,05	2,8273
7	1,5 - 10 - 0,5 - 1	150,74	2,404
8	1,5 - 10 - 0,75 - 1	175,54	2,6395
9	1,4 - 8 - 0,5 - 1,5	141,25	2,1214
10	1,4 - 8 - 0,75 - 1,5	143,24	2,2544
11	1,4 - 10 - 0,5 - 1,5	142,54	2,1427
12	1,4 - 10 - 0,75 - 1,5	150,37	2,2063
13	1,5 - 8 - 0,5 - 1,5	141,59	2,2122
14	1,5 - 8 - 0,75 - 1,5	154,53	2,2764
15	1,5 - 10 - 0,5 - 1,5	143,14	2,0202
16	1,5 - 10 - 0,75 - 1,5	172,02	2,1905

Tabla 4. Esfuerzo máximo del hueso cortical y trabecular para diferentes configuraciones. En la columna "modelo", los valores hacen referencia a diámetro, longitud roscada, paso y espesor de hueso cortical respectivamente. Nota: valores en mm

La distribución de esfuerzo de cada uno de los modelos se pueden observar en la [Figura 3], en donde se organizaron de tal forma que se pueda diferenciar claramente como las variaciones en las características geométricas y el espesor de hueso cortical, afectan la carga sobre el hueso. Se evidencia que los pasos pequeños y las longitudes roscadas largas inciden sustancialmente en la distribución de esfuerzos.

Con el fin de conocer el comportamiento de los esfuerzos en la zona ósea bajo la variación del

diámetro, paso de rosca, longitud roscada y espesor de hueso cortical, se realizó una toma de valores de esfuerzo a lo largo del espesor del hueso cortical y la longitud del hueso trabecular [Figura 3(1)]. Gracias a ello se obtiene una serie de graficas en donde se busca comparar la variación de esfuerzos al variar alguna de las características. En el primer conjunto de gráficas [Figura 4] se busca ver la afectación que sufre la interfaz hueso –MI frente a la variación del paso de rosca y el espesor de hueso cortical manteniendo un diámetro de 1,4mm.

Para el siguiente conjunto [Figura 5] el diámetro se varía 0,1mm y al igual que el anterior, las variaciones se realizan en el paso de rosca y el espesor de hueso cortical. El tercer grupo de gráficas [Figura 6] permite observar la variación de los esfuerzos al mantener el paso de rosca y espesor de cortical contantes, variando el diámetro y la longitud roscada. Finalmente en la [Figura 7] al igual que en el grupo anterior se varía el diámetro y la longitud roscada pero se aumenta el espesor de hueso cortical en 0,5mm.

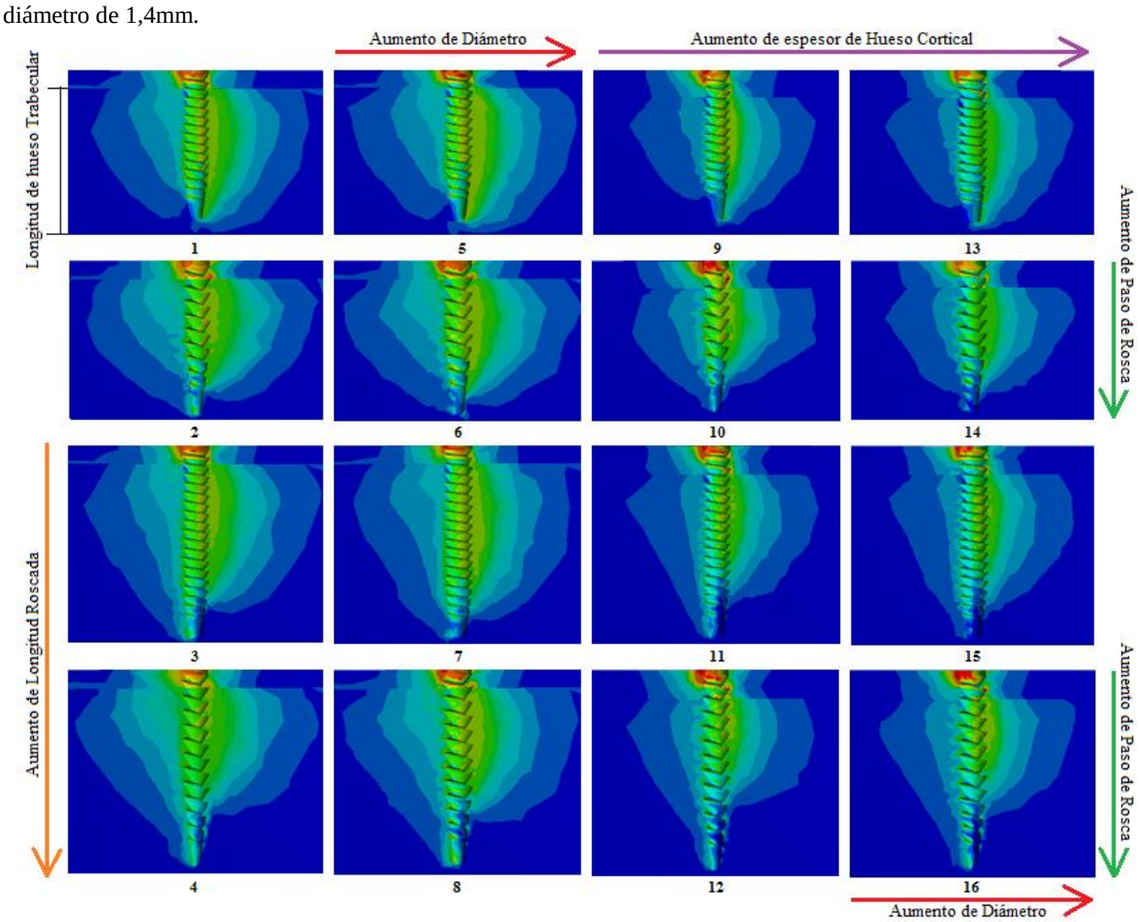


Figura 3. Distribución de esfuerzos del hueso cortical y trabecular para las 16 configuraciones bajo la variación de diámetro, longitud roscada, paso de rosca y espesor de hueso cortical, las flechas indican un aumento en las variables respecto a las filas o columnas anteriores. La numeración está acorde a las dimensiones de la columna "modelo" establecidas en la [Tabla 4].

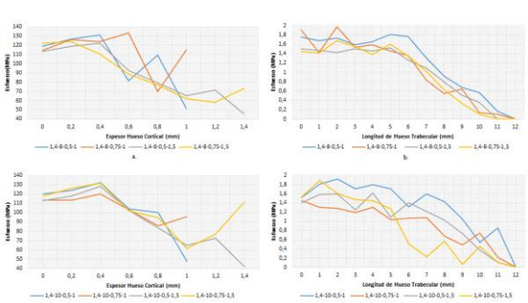


Figura 4. Distribución de esfuerzos a lo largo del hueso cortical y trabecular al variar paso y espesor de cortical. a. y b. Valores para el hueso cortical y trabecular para una configuración 1,4mm Ø – 8mm longitud roscada, variando paso de rosca y espesor de hueso cortical. c. y d. Valores para el hueso cortical y trabecular para una configuración 1,4 mm Ø – 10mm longitud roscada, variando paso de rosca y espesor de hueso cortical.

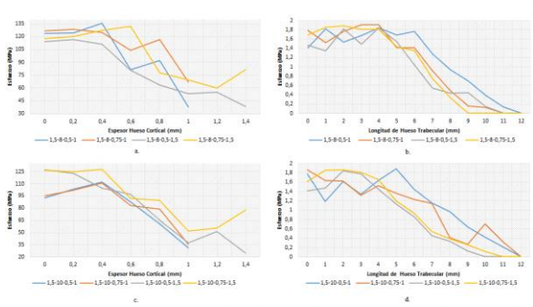


Figura 5. Distribución de esfuerzos a lo largo del hueso cortical y trabecular al variar paso y espesor de cortical. a. y b. Valores para el hueso cortical y trabecular para una configuración 1,5 mm Ø – 8mm longitud roscada, variando paso de rosca y espesor de hueso cortical. c. y d. Valores para el hueso cortical y trabecular para una configuración 1,5 mm Ø – 10mm longitud roscada, variando paso de rosca y espesor de hueso cortical.

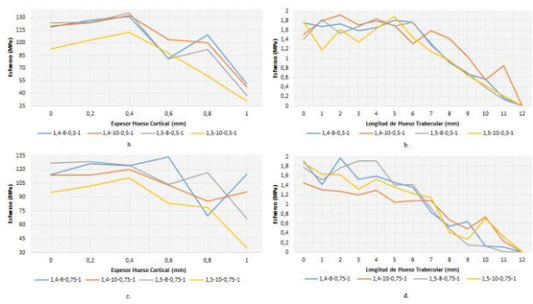


Figura 6. Distribución de esfuerzos a lo largo del hueso cortical y trabecular al variar diámetro y longitud roscada. a. y b. Valores para el hueso cortical y trabecular para una configuración 0,5mm paso de rosca – 1mm espesor de cortical, variando diámetro y longitud roscada. c. y d. Valores para el hueso cortical y trabecular para una configuración 0,75mm paso de rosca – 1mm espesor de cortical, variando diámetro y longitud roscada.

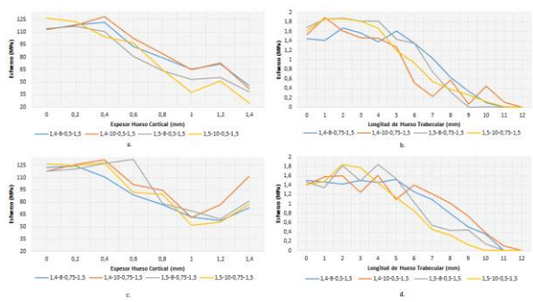


Figura 7. Distribución de esfuerzos a lo largo del hueso cortical y trabecular al variar diámetro y longitud roscada. a. y b. Valores para el hueso cortical y trabecular para una configuración 0,5mm paso de rosca – 1,5mm espesor de cortical, variando diámetro y longitud roscada. c. y d. Valores para el hueso cortical y trabecular para una configuración 0,75mm paso de rosca – 1,5mm espesor de cortical, variando diámetro y longitud roscada.

Discusión

Como se puede observar en la [Figura 3], la distribución de esfuerzos sobre las zonas de hueso cortical es similar, para esta zona el valor máximo fue de 202,58 MPa, reportada en un implante corto de paso largo. Aquellos modelos con un espesor de hueso cortical de 1,5mm presentan valores de esfuerzo máximo menores a los de espesor de 1mm, siendo este el caso del modelo #5 y #7 de la [Tabla 4] en donde se presenta una disminución de 3,09% para el hueso cortical. Su distribución de esfuerzo va totalmente ligada al paso de rosca que posee el mini-implante.

La zona trabecular reporta un mayor cambio en su distribución de esfuerzos que en sus valores máximos, los resultados demuestran que los implantes de longitud mayor distribuyen más eficientemente el esfuerzo y de mejor manera mediante pasos menores. Comparando el modelo #3 y #4 de la [Tabla 4] se observa el notable cambio sufrido en la distribución de esfuerzos de la zona trabecular al variar 0,25mm el paso de rosca, para este caso existe un aumento del 8,58% en el hueso cortical y una disminución de 3,13% en el hueso trabecular (porcentajes referidos frente a valores máximos). Ahora bien, en el modelo #3 y #7 de la [Tabla 4] en donde la variable geométrica alterada es el diámetro, se observa que un aumento de 0,1mm en el diámetro genera un aumento de esfuerzo máximo en 2,11% en el hueso cortical y una disminución de 9,28% en el hueso trabecular.

El comportamiento de los esfuerzos a lo largo del hueso determina la calidad de estabilidad primaria que el implante poseerá en su interfaz, estos estudios permitieron comparar cual es el porcentaje de afectación de cada una de las variables y así conocer cuál de ellas es la que afecta en mayor o menor medida el comportamiento de los esfuerzos y deformaciones del conjunto MI – hueso.

En la [Figura 4a y 4c] se observa que la distribución de esfuerzos en la zona cortical para un implante con un paso de 0,75mm, experimenta valores finales crecientes, mientras que un implante con paso de 0,5mm muestra una distribución uniforme que siempre decrece. Esta observación es igual al variar el espesor de hueso cortical de 1mm a 1,5mm y observar el comportamiento de las gráficas respecto al paso de rosca. La zona trabecular correspondiente a la [Figura 4b y 4d] experimenta una distribución homogénea con valores altos decrecientes para modelos con valores de paso de 0,5mm, lo cual es un indicio de una buena traba mecánica dado que la totalidad del área de contacto de la rosca está interactuando frente a la carga impuesta. Estos resultados son acordes al estudio realizado por [Cunha, 2015] en donde se muestra que para MIs con paso de rosca mayores los torques de desinserción disminuyen.

La [Figura 5] utiliza las mismas variaciones en cuanto a paso de rosca y espesor de cortical que la figura anterior, sin embargo el análisis se hace con un diámetro 0,1mm mayor al anterior, lo cual genera una disminución en los valores de esfuerzo a lo largo de todo el espesor de hueso cortical y trabecular. La respuesta a la variación de paso de rosca y espesor de hueso cortical experimenta el mismo comportamiento de las gráficas de la [Figura 4]. Dada la distribución de esfuerzos y el valor de los esfuerzos máximos es aconsejable que se trabaje con la inserción de MIs de paso corto, ya que ofrecen una rigidez mayor y una distribución homogénea de la carga a lo largo de toda la zona roscada.

La longitud roscada y el diámetro son evaluadas en la [Figura 6] y la [Figura 7], los resultados de las gráficas muestran que cuando la longitud roscada aumenta 2mm no se experimentan cambios significativos en la distribución de esfuerzos a lo largo del hueso cortical, sin embargo para el hueso trabecular la distribución de esfuerzos es similar hasta los 8mm de longitud, de ahí en adelante los valores crecen para los modelos con 10mm de longitud roscada, lo cual indica que la zona extra de agarre se encuentra sometida bajo esfuerzos.

Para observar el comportamiento de la variación del diámetro, se tomará la [Figura 7], en ella se mantienen constantes el paso de rosca y el espesor de hueso cortical. Los resultados muestran que el aumento del diámetro provoca valores máximos mayores para la zona cortical, creciendo en un rango de 1,57% a 3,3% respecto a los modelos que no varían su diámetro. Este resultado muestra que es beneficioso aumentar el diámetro dado que los esfuerzos se distribuyen con valores menores a lo largo del hueso [Figura 3(6)] y [Figura 7a y 7b], sin embargo hay que tener en cuenta que la inserción de diámetros más grandes requiere cavidades previas mayores, lo cual como se muestra en [Okazaki, 2008], genera torques de remoción instantáneos menores y un maltrato a nivel general del hueso, que no son beneficiosos en la búsqueda de una buena estabilidad primaria.

En cuanto al espesor de hueso cortical, el análisis se hace en la [Figura 4] y [Figura 5], las gráficas muestran como resultado que los modelos con un espesor de cortical de 1,5mm respecto a los de 1mm presentan una disminución en los valores de esfuerzo a lo largo del espesor de hueso cortical siempre y cuando el paso del implante sea corto, cuando el implante es de paso largo se presentan mayores valores de esfuerzo al finalizar la capa cortical. El espesor de hueso cortical afecta en gran medida la distribución de esfuerzos de la zona trabecular como se muestra en la [Figura 3(3) y 3(11)] y el valor de los esfuerzos a lo largo de la longitud del hueso trabecular [Figura 5b y 5c] disminuye entre un 6,39% 19,19% cuando se usa MIs de paso corto.

Este estudio a diferencia de otros donde solo se analiza una variable, permitió determinar la distribución de esfuerzos en función de diferentes variables geométricas y biológicas (espesor de hueso cortical) y como éstas interactúan entre sí. Se concluye que entre menor sea el paso de rosca la distribución de esfuerzos se comportará de manera más uniforme sobre toda la interfaz y que los valores de sus esfuerzos pueden ser menores cuando se presentan mayores espesores de hueso cortical. En ausencia de hueso cortical se aconseja el uso de MIs de gran longitud roscada y diámetros mayores, dado que estas configuraciones permiten una mejor rigidez al tener más área de contacto en el hueso trabecular. El conocimiento de la variación de esfuerzos y deformaciones frente a las distintas características geométricas de los MI y las condiciones biológicas, permite aconsejar cual dispositivo de anclaje temporal debería usarse sobre un paciente con determinadas condiciones físicas para que el porcentaje de falla de éste debido a condiciones mecánicas se reduzca considerablemente.

Referencias

Ammar, H. H., Ngan, P., Crout, R. J., Mucino, V. H., & Mukdadi, O. M. (2011). Three-dimensional modeling and finite element analysis in treatment planning for orthodontic tooth movement. *American Journal of Orthodontics and Dentofacial Orthopedics*, *139*(1), e59–e71.

Buschang, P., Carrillo R, B, O., & PE, R. (2008). Survey of AAO members on miniscrew usage. *J Clin Orthod*, 513–8.

Cunha, A. C. Da, Freitas, A. O. A. De, Marquezan, M., & Nojima, L. I. (2015). Mechanical influence of thread pitch on orthodontic mini-implant stability. *Brazilian Oral Research*, *29*(1), 1–6.

Cunha, A. C. Da, Marquezan, M., Lima, I., Lopes, R. T., Issamu Nojima, L., & Franzotti Sant'anna, E. (2015). Influence of bone architecture on the primary stability of different mini-implant designs. *American Journal of Orthodontics and Dentofacial Orthopedics*, *147*(1), 45–51.

Gainsforth, B., & Higley, L. (1945). A study of orthodontic anchorage possibilities in basal bone. *Am J Orthod Oral Surg*, *31*, 406–517.

Kong, L., Liu, B., Li, D., Song, Y., Zhang, A., Dang, F., … Yang, J. (2006). Comparative study of 12 thread shapes of dental implant designs : a three-dimensional finite element analysis. *World Jorunal of Modelling and Simulation*, *2*(2), 134–140.

Martínez-González, J. M., Cano Sánchez, J., Campo Trapero, J., Martínez-González, M. J. S., & García-Sabán, F. (2002). Diseño de los implantes dentales: Estado actual. *Avances En Periodoncia E Implantología Oral*, *14*(3).

Okazaki, J., Komasa, Y., Sakai, D., Kamada, A., Ikeo, T., Toda, I., … Etoh, T. (2008). A torque removal study on the primary stability of orthodontic titanium screw mini-implants in the cortical bone of dog femurs. *International Journal of Oral and Maxillofacial Surgery*, *37*(7), 647–650.

Reynders, R., Ronchi, L., & Bipat, S. (2009). Mini-implants in orthodontics: A systematic review of the literature. *American Journal of Orthodontics and Dentofacial Orthopedics*, *135*(5), 564.e1-564.e19.