ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL PARA UNA MÁQUINA DE CONTROL NUMÉRICO COMPUTARIZADO (CNC) SOBRE UN SISTEMA EMBEBIDO UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE"

INFORME DE PROYECTO DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN CIENCIAS COMPUTACIONALES

ESPECIALIZACIÓN SISTEMAS MULTIMEDIA

Presentado por

CARLOS ALBERTO RONQUILLO CASTRO

GUAYAQUIL - ECUADOR

2015

AGRADECIMIENTO

Agradezco en primer lugar a mi familia por su apoyo incondicional.

Agradezco a todo el equipo del Centro de Visión y Robótica de la ESPOL donde pude involucrarme en este proyecto, al Ph.D. Daniel Ochoa por darme la oportunidad de ser parte de ese grupo, a mi director de Proyecto MSc. Jorge Magallanes por su guía y colaboración durante todo el proceso.

Carlos Ronquillo Castro

DEDICATORIA

Dedico este trabajo a mi padre y a mi madre quienes han sido un pilar fundamental por todo lo que han sacrificado para que pueda llegar a esta etapa de mi vida y lograr esta meta.

Carlos Ronquillo Castro

TRIBUNAL DE SUSTENTACIÓN

SUBDECANO DE LA FIEC

PRESIDENTE DEL TRIBUNAL

MSc. Jorge Magallanes B.

DIRECTOR DEL PROYECTO DE GRADUACIÓN

Ph.D. Xavier Ochoa Chehab

MIEMBRO DEL TRIBUNAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Informe, me corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral"

(Art. 12 Reglamento de Graduación de la ESPOL)

Carlos Ronquillo Castro

RESUMEN

El siguiente trabajo muestra la implementación de un Sistema de Control para una máquina CNC (Control Numérico Computarizado) utilizando únicamente Software Libre e implementándolo en un Sistema Embebido. Como prototipo experimental para las pruebas se utilizó una impresora 3D en reemplazo de la maquinaria típica en este tipo de Sistemas, tales como fresadoras y tornos, ya que cuenta con similares características en su estructura de ejes y motores.

El proyecto abordó dos de las etapas que conforman el proceso de manufactura en un Sistema CNC. Primero, utilizando una herramienta CAM (Computer-Aided Manufacturing) se generó del código que detalla la trayectoria que deberá seguir la herramienta de la máquina, todo en base a un diseño 3D CAD (Computer-Aided Design). La segunda etapa consiste en la ejecución de este código utilizando el Software de Control CNC para generar las señales transmitidas desde el Sistema Embebido a los motores de la máquina vía puerto paralelo. También, se diseñó un controlador especializado para aprovechar el uso de pines de propósito general (GPIO) como una alternativa al puerto paralelo.

Se realizaron varias pruebas para asegurarse de la correcta configuración de la máquina, entre ellas el monitoreo de las señales de salida del puerto paralelo, la ejecución de código, y además de pruebas individuales a los motores de la máquina. Al final se consiguió tener un Sistema de Control preparado para ser utilizado junto a diferentes tipos de maquinaria CNC.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

CAD: Computer-Aided Design

CAM: Computer-Aided Manufacturing

CNC: Control Numérico Computarizado

GPI: Entrada de Propósito General

GPIO: Entrada/Salida de Propósito General

GPO: Salida de Propósito General

HAL: Capa de Abstracción de Hardware

RTAI: Interfaz de Aplicación en Tiempo Real

MDI: Ingreso Manual de Datos

PMIO: Port-Mapped Input/Output

INTRODUCCIÓN

En la actualidad países que basan su economía en recursos naturales como Ecuador pueden garantizar una buena calidad de vida a sus habitantes, sin embargo estos recursos no durarán para siempre, por eso es necesario establecer una economía basada en la Industria, ofreciendo una gran cantidad de bienes y servicios que otros países requieran comprar.

Una forma de lograrlo es mejorando la eficiencia y calidad de nuestros métodos de manufactura, por ejemplo la automatización de maquinaria usando Control Numérico Computarizado (CNC), este proyecto busca demostrar que se puede crear un Sistema de Control de bajo costo con el que cualquier empresario en el área de manufactura pueda mejorar la calidad de su producción y competir con otros mercados.

En el Capítulo 1 se define el problema que estamos tratando de resolver y la justificación de la realización de este proyecto, cuales son los objetivos generales y específicos que se tratan de alcanzar y la metodología que seguiremos.

El Capítulo 2 introduce el concepto básico del Control Numérico Computarizado, los elementos que conforman el Sistema CNC, el proceso de

diseño y una descripción de las herramientas y equipo utilizado en el proyecto.

En el Capítulo 3 se explica la implementación y configuración del Sistema utilizando las herramientas de Software Libre.

En Capítulo 4 se detallan los requerimientos del Sistema y su arquitectura, mostrando una visión general del Sistema y los componentes que lo conforman.

En el Capítulo 5 se muestran los resultados obtenidos en las pruebas de funcionamiento del Sistema de Control utilizando como máquina CNC una impresora 3D.

CAPÍTULO 1

1. ANTECEDENTES

1.1 Descripción del Problema

En la actualidad, en la Industria Manufacturera, para mantenerse en el mercado competitivo, los fabricantes deben realizar ajustes de carácter tecnológico e invertir en tecnología especializada para la manufactura. Un ejemplo de este tipo de tecnología son las máquinas de control numérico (CNC) para elaborar productos por medio del tallado o desbaste de material.

Sin embargo, a pesar de tener un gran mercado potencial, en Ecuador son muy escasas las empresas dedicadas a la construcción de este tipo de maquinaria. Esto debido a la falta de fuentes de inversión nacional, y por otro lado a que existe la noción de que lo producido nacionalmente no se iguala a lo importado.

1.2 Justificación

En Ecuador la producción nacional de maquinaria especializada para manufactura no es muy competitiva a nivel mundial, esto ha llevado a la Industria Manufacturera a buscar alternativas que le permitan mejorar la calidad de sus productos, como la importación de maquinaria especializada en automatización industrial o contratar terceros para que se encarguen de la comercialización y capacitación de la tecnología. Por otro lado, se encuentran las pequeñas y medianas empresas que no cuentan con los recursos económicos para invertir e importar esta maquinaria.

Esta propuesta busca apoyar a la producción nacional y a estas empresas, de manera que, se desarrolle localmente a bajo costo una maquinaria de control numérico computarizado (CNC), especializada para la manufactura en metal y madera que les permita la automatización de sus líneas de producción.

La inversión e implementación de esta tecnología por parte de empresas nacionales en la Industria Manufacturera para automatizar su maquinaria les permitirá alcanzar un nivel más alto de competitividad, mejorando la relación calidad-precio de su producción

al poder obtener acabados más finos y detallados con gran precisión y disminuyendo tiempos y costos.

1.3 Solución Propuesta

La solución planteada en este reporte es la implementación de un Sistema de control CNC basada en PC, se utilizara una plataforma embebida para que esté acorde a las características de un PC industrial y software libre con una herramienta CAM para obtener el código G con las instrucciones para la máquina y el software de control CNC.

Una de las ventajas de utilizar software libre a más de reducir considerablemente el costo total del producto, es poder adaptarlo a nuestra conveniencia, por eso se desarrolló un controlador para el uso de pines GPIO para el envío de señales de motores, sensores o interruptores como una alternativa adicional al puerto paralelo utilizado normalmente y para mejorar las características del sistema.

1.4 Objetivos

Objetivo General: Implementar un Sistema de Control de una Máquina CNC en un Sistema Embebido utilizando herramientas de Software Libre, que permita a la Industria Manufacturera Ecuatoriana mejorar su maquinaria y aumentar la calidad de la producción nacional.

Objetivos Específicos

- Implementar el Sistema en un hardware embebido con Sistema Operativo Linux y el Software de Control de Maquinaria LinuxCNC.
- Establecer la configuración adecuada del Sistema que permita adaptar los diferentes tipos de maquinaria CNC de acuerdo a sus características.
- Diseñar e implementar un controlador especializado que aproveche las características del sistema embebido, adaptándolo para que ofrezca más opciones de control y operación al Sistema.

1.5 Metodología

El Sistema se implementará en un Sistema embebido Advantech PCM-9562, al que se le instalará una versión de Linux con extensiones de real-time utilizando un parche RTAI, además del Software de Control LinuxCNC.

Como prototipo de una máquina CNC se utilizará una impresora 3D, ya que utiliza los mismos tipos de motores de paso, los cuales serán controlados mediante señales del puerto paralelo del embebido.

Se realizarán pruebas en los motores, determinando la velocidad y aceleración máxima en la que funcionen correctamente y se evaluará el funcionamiento del sistema durante la ejecución del código G obtenido de un modelo 3D mediante la herramienta PyCAM.

Se implementará el mismo Sistema en un embebido VIA EPIA-P830 para demostrar la posibilidad de utilizar pines GPIO como opciones de control en el sistema mediante el diseño de un controlador especializado.

CAPÍTULO 2

2. MARCO TEÓRICO

2.1 CNC

2.1.1 Definición

Control Numérico Computarizado (CNC) describe la automatización de Máquinas-Herramientas, es decir, una computadora controla los movimientos de los ejes de una máquina mediante un código el cual contiene instrucciones secuenciales de los movimientos de las herramientas, giros, las velocidades de corte, puntos de inicio y fin de las operaciones. Este código permite crear piezas y partes tallándolas a partir de un bloque de material cada vez que es ejecutado [1].

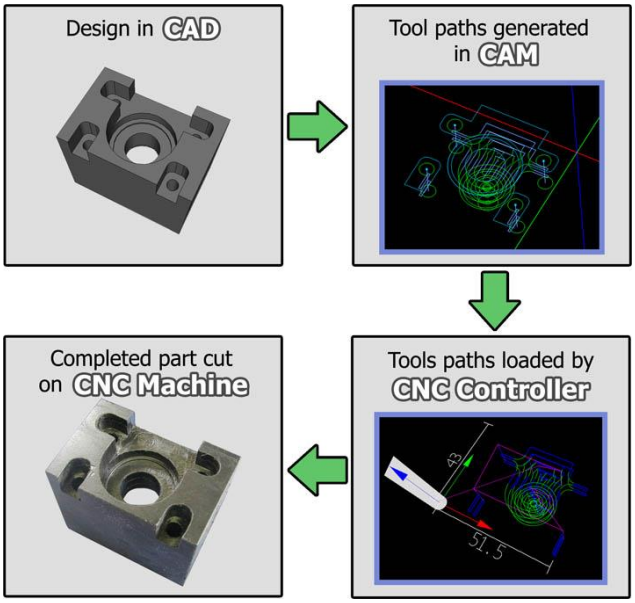


Figura 2.1 *Proceso de Manufactura CNC. [2]*

El proceso de diseño a proyecto finalizado en CNC puede resumirse como se aprecia en la Figura 2.1, todo empieza con la idea plasmada en un diseño mediante una Herramienta CAD (Computer-Aided Design), el cual contiene todos los detalles de su geometría y dimensiones, luego este diseño es convertido en datos para controlar a la máquina mediante una herramienta CAM (Computer-Aided Manufacturing), en donde se definirán las estrategias a seguir para cortar el material y los tipos de herramientas necesarias para obtener la pieza final, todo esto es almacenado en un archivo de código-G, la siguiente etapa interviene el Software de Control CNC, este interpretara los comandos del código-G y los transmitirá en forma de señales que

la máquina-herramienta pueda entender para el movimiento de sus motores y herramientas, y así se finaliza con el producto o pieza terminada. [2] [3]

2.1.2 Máquinas CNC

Las máquinas CNC surgieron como medida para superar las limitaciones que se tenían con maquinaria manual, como como piezas con geometrías complejas, utilizando un computador para calcular las trayectorias y manejar la máquina mediante control numérico se pueden crear piezas con gran velocidad y precisión. Es de notar que cualquier máquina-herramienta convencional como tornos y fresadoras pueden adaptarse para trabajar con CNC.

2.1.2.1 Características de una Máquina CNC

En la actualidad hay disponible una gran diversidad de modelos de máquinas CNC, cada una con variaciones dependiendo de los requerimientos de los usuarios, algunas de las características más importantes se detallan a continuación: [4]

Características del Sistema de Control

- Tecnología del Sistema: Un procesador o multi-procesador
- Calibración de Ejes

- Modos de operación: Manual (MDI), paso a paso, automático.
- Controles de operación: Movimiento, feedrate, velocidad del husillo, cambio de ejes.
- Compensación de Errores: Inclinación, Inversión, Herramientas

Características de Programación

- Subrutinas y Macros: Para operaciones repetitivas
- Programación paramétrica, para operaciones aritméticas en el código-G.
- Ciclos Fijos, ayudan a conservar memoria reduciendo el tamaño del código.
- Diagnóstico: Programas que monitorean el funcionamiento del sistema CNC en varios niveles y pueden trabajar en tiempo real.

Otras Características Avanzadas

- Determinación automática del toolpath óptimo.
- Cambio de orientación del cabezal de forma rápida y precisa.
- Feedrate automático, en especial en curvas y esquinas.
- Corte a alta velocidad de contornos complejos

- Detección de Posición Absoluta, para que no sea necesario que la máquina regrese a su posición inicial después de finalizar una operación.
- Medición automática de la longitud de herramientas.

2.1.2.2 Elementos de una Máquina CNC

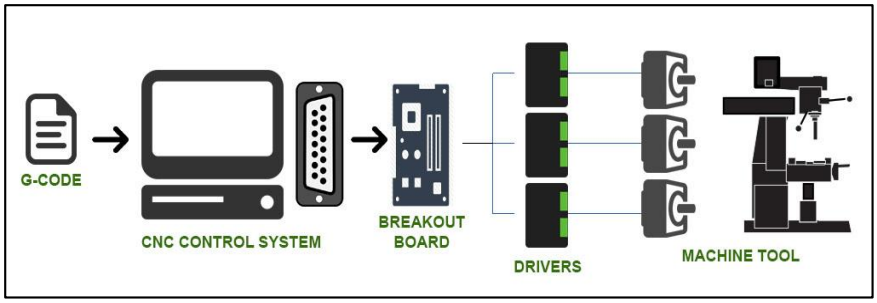


Figura 2.2 *Elementos de un Sistema CNC basado en PC*

Los elementos más comunes que conforman una Máquina CNC se muestran en la Figura 2.2 y son:

Sistema de Control CNC

Es el dispositivo electrónico en las máquinas CNC especializadas, cuya función es controlar tanto los motores de los ejes como el de la herramienta, también se lo conoce como (Machine Control Unit) [5].

Hoy en día se pueden construir Sistemas de bajo costo a base de Computadoras Personales, Software de Control, esto permite a un alto rango de usuarios experimentados o

aficionados producir sus propias partes y piezas con facilidad [3].

Las características básicas que deben tener los PC para este propósito incluyen:

- **Procesador multinúcleo**: Para garantizar la rapidez del cálculo de trayectorias y el envío de señales.
- **RAM** de alta capacidad en caso de utilizar las herramientas CAD/CAM.
- **Sistema Operativo de Tiempo Real**, esta es una característica importante ya que el envío de señales debe ser en tiempo real para garantizar su precisión, en PC se suele utilizar Windows y Linux junto a extensiones real-time ya que normalmente no tienen esta funcionalidad, un ejemplo es RTAI en el caso de Linux.

Software de Control CNC

Este software va a ser el encargado de recibir los diseños, representados en código-G y obtenidos mediante herramientas CAM, procesarlos y convertirlos en señales que serán enviadas a la tarjeta interfaz y luego controlaran la máquina CNC. Las opciones más populares que se encuentran disponibles son Mach3 y LinuxCNC.

- **Mach3**

  Mach3 es uno de los más populares controladores CNC, es un software de pago compatible con una gran variedad de dispositivos, además de que ofrece la posibilidad de expandirlo con paquetes opcionales, agregándole funcionalidad CAM para que pueda crear su propio código-G. Mach3 trabaja en Sistemas Operativos Microsoft Windows, entre sus características destacan poder personalizar completamente la interfaz y el uso de Macros de Visual Basic.

- **LinuxCNC**

  LinuxCNC antiguamente conocido como EMC2 es otra alternativa, es software libre y la opción que se ha usado en este proyecto y la que se detallara más adelante. Trabaja en Sistemas Operativos Linux, entre sus características destacan el uso de una capa de abstracción de Hardware (HAL), un osciloscopio incorporado, la capacidad de operar hasta 9 ejes simultáneamente [6].

Hardware de Control CNC

Controlador de motor de paso

Los controladores son los dispositivos que manejan a los actuadores recibiendo las señales y convirtiéndolas en movimiento a través de los motores de paso o servo motores [7]. Ya que están conectados directamente a los motores, es necesario uno por cada motor.

Los controladores de paso pueden amplificar la tasa de frecuencia de las señales que reciben de la tarjeta interfaz mediante micro-pasos, por ejemplo, en un motor con la capacidad de 200 pasos por revolución, pueden convertir el paso original obtenido en 20 micro pasos, obteniendo 4000 pasos por revolución lo que aumenta la resolución del motor. Durante la operación normal de estos dispositivos suelen producir grandes cantidades de calor por lo que requieren de disipadores para un correcto funcionamiento [6].

Tarjeta de Interfaz o Breakout Boards

Las tarjetas Interfaz trabajan como un nexo entre la PC con el Software de Control y los controladores de los motores, su tarea consiste en aislar las señales entre ellos y mantener el PC seguro, también pueden proveer conexiones para sensores, switches, y otros dispositivos. [7].

Estas tarjetas se conectan al PC mediante el puerto paralelo, el cual es uno de las más utilizados en aplicaciones CNC, y como ya se mencionó, una de sus funciones principales es una especie de aislamiento eléctrico entre el computador y el controlador del motor, evitando que cualquier daño pueda regresar al PC por una falla en el cableado o un alto voltaje.

Las tarjetas de interfaz pueden tener variedad de formas y tamaños, pueden ser diseñadas para soportar de 3 a 5 motores además de aislar señales de hasta 2 puertos paralelos [6].

Actuadores

Los principales actuadores usados en Máquinas CNC son los motores de paso y servo motores.

Motor de Paso

Los motores de paso destacan por ser de alta precisión, lo que los hace ideales para aplicaciones que requieran movimiento con exactitud y velocidad como las Máquinas CNC. Estos motores poseen un alto número de polos magnéticos dentro del bobinado del estator, esto permite al rotor lograr giros pequeños.

Al recibir una señal el motor gira en un pequeño ángulo y espera por la siguiente señal lo que se considera un paso, normalmente el giro es de 1.8°, por lo que serían 200 pasos para realizar una revolución completa. Entre mayor cantidad de pasos, más fino es el movimiento del motor, esto se puede lograr usando controladores de micro-pasos que pueden dividir el paso original en 10 micro-pasos por ejemplo y obtener 2000 pasos por revolución [3] [6].

Servo Motor

Un servo motor es bastante parecido a un motor de paso y produce los mismos resultados, pero utiliza otra técnica, el servo motor tiene incorporado un encoder, este dispositivo monitorea la posición actual del motor en tiempo real y la

corrige enviando señales de error en caso de que detecte que el eje del motor no se encuentra en su posición objetivo, esta es una de sus ventajas ya que produce retroalimentación de su estado al software de control CNC, algo que los motores de paso no pueden hacer sin un encoder externo.

Esta característica y su gran precisión los hace ser los más utilizados en la Industria, además de ser más costosos que los motores de paso [3] [6].

2.1.2.3 Tipos de Máquinas CNC

Las máquinas CNC se pueden dividir de forma básica en dos grupos: Tornos y Fresadoras. Sin embargo cualquier máquina-herramienta que pueda ser controlada manualmente puede ser adaptada para usar controles CNC [8].

Fresadora

La fresadora es la máquina-herramienta más común en la industria, consiste en una herramienta rotatoria cortante llamada fresa, la cual se mueve de forma vertical cortando y tallando el material a lo largo de su superficie mediante una base móvil. Las fresadoras más comunes se mueven en 3 dimensiones sin

embargo se puede añadir una cuarta o quinta dimensión mediante un cabezal o base rotatoria [8].

Torno

Los tornos son otro tipo de máquina-herramienta, estas giran el material a una alta velocidad alrededor de un husillo mientras que una herramienta cortante la cual se puede mover en 2 ejes X y Z, se acerca para tallar la pieza. Debido a estas características los tornos son ideales para el tallado de piezas que son simétricas alrededor de un eje [9].

Enrutador (Router)

Los enrutadores son máquinas-herramientas específicamente diseñadas para ser usadas mediante CNC, en realidad son un tipo de fresadora, pero están enfocadas en trabajar a altas velocidades y con materiales ligeros como madera, plásticos e incluso finas láminas de aluminio. Por su versatilidad están ganando terreno rápidamente en la industria [8].

CNC Cortador Laser y Cortador de Plasma

Estas son máquinas que se mueven en 2 dimensiones, son utilizadas principalmente en la Industria Metalúrgica, para cortes de láminas y placas de metal con alta precisión utilizando laser

de alta energía o plasma. Los cortadores laser son bastante rápidos y precisos logrando una excelente calidad en cualquier patrón de corte imaginable, mientras que los cortadores de plasma están destinados a trabajo más pesado y su corte suele ser algo irregular y no tan preciso debido a la naturaleza de la flama de gas ionizado [8].

Impresoras 3D y Máquinas de Prototipado Rápido

Este tipo de máquinas utiliza una configuración similar a las fresadoras y enrutadores, la diferencia es que en vez de tallar las piezas con herramientas de corte, utilizan una extrusora de plástico caliente la cual lentamente libera el plástico capa por capa hasta crear la pieza completa [10].

2.1.2.4 Herramientas de Corte

Las Fresadoras, Tornos o Enrutadores, utilizan una gran variedad de herramientas de corte ubicadas en el cabezal de la máquina, existen de diferentes formas: cilíndricas, circulares, de disco, y algunas se especializan en fases de remover material, contorneado y acabados, las más comunes son las siguientes:

Figura 2.3 *Brocas y Escariadores*

Para crear agujeros:

Comúnmente se hacen utilizando Brocas helicoidales, Escariadores (ver Figura 2.3), pero mediante máquinas CNC también es posible mediante fresas de varios diámetros.

Para crear superficies planas:

Fresas, taladros, cortadoras de ranura de cola de milano, cortadores de ranura en T, sierras de corte longitudinal son algunas de las más comunes, en la Figura 2.4 se muestran algunas.

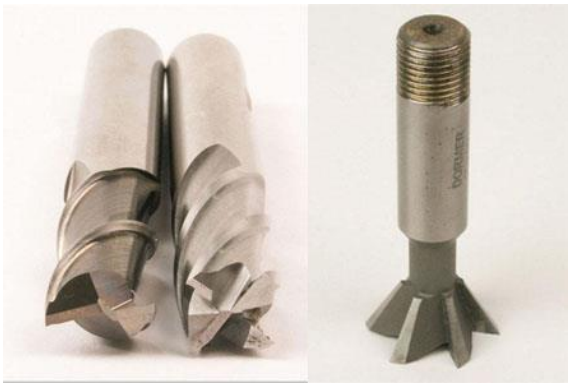


Figura 2.4 *Taladro, Fresa y Herr. Cola de Milano. [3]*

Para crear Formas:

Aunque con el uso de CNC ya no son muy necesarias ya que se puede lograr el mismo resultado con herramientas convencionales, aun son utilizadas para facilitar esa tarea. Estas son Fresas de punta esférica, cortador para redondeado de esquinas, cortadores cóncavos y convexos.

2.2 Herramientas CAD/CAM

El uso de las Herramientas CAD/CAM comprende la fase inicial del proceso de diseño y manufactura como se muestra en la Figura 2.5, describiremos cada una a continuación.

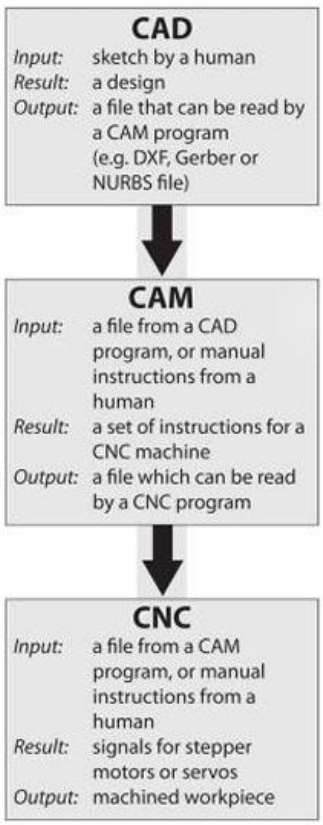


Figura 2.5 *Etapas del Proceso CAD/CAM/CNC. [3]*

2.2.1 CAD

CAD (Computer-Aided Design) es el proceso de diseño de una parte o pieza a elaborar, mediante vectores. El diseño CAD constituye la fase inicial del proceso de manufactura CNC (ver Figura 2.5), donde la idea o el boceto de una persona es plasmado y representado de forma gráfica a escala real y con todas sus características geométricas [6].

Las herramientas CAD permiten tanto realizar diseños 2D con facilidades como agregar texto lo que es muy útil para el grabado de materiales, y también diseño 3D con el que se pueden dibujar complicadas estructuras y piezas en una fracción de tiempo de lo que llevaría dibujarlo en papel, además de contar con la opción de realizar cualquier modificación sin tener que empezar un nuevo diseño.

Existen innumerables opciones al elegir herramientas CAD, las comerciales, software libre, desde las más básicas a las más complejas y con mayores funcionalidades, entre ellas están: AutoCAD y AutoCAD LT, TurboCAD, QCAD, también existen herramientas graficas que trabajan con el formato vectorial como Adobe Illustrator, GIMP, Rhino, entre otras.

2.2.2 CAM

CAM (Computer-Aided Manufacturing) constituye la segunda etapa en el proceso de manufactura CNC (ver Figura 2.5), Las herramientas CAM básicamente permiten importar diseños vectoriales CAD, y convertirlos en una secuencia de códigos y funciones que puedan ser interpretados por el Sistema CNC, por

ejemplo código-G, para que la máquina-herramienta realice las tareas necesarias para elaborar la parte o pieza diseñada [11].

El usuario puede también producir código-G por su cuenta, especificando paso por paso las operaciones que realizara la máquina para obtener el mismo resultado, luego ingresando estos comandos de forma manual utilizando el Sistema de Control de la máquina, sin embargo cuando la complejidad de la pieza aumenta, esta puede requerir de una gran cantidad de líneas de código, para ello fueron desarrolladas las herramientas CAM.

Generalmente el proceso de creación de un código-G mediante herramientas CAM es el siguiente:

- Crear un modelo de producción a partir del archivo de diseño CAD, y la pieza de trabajo o material (metal, madera, plástico).

- Establecer una base de datos de herramientas, aquí deben estar definidas todas las herramientas que pueden ser utilizadas por la máquina-herramienta o las que serán

utilizadas en este diseño en particular. La información que debe contener va desde el tipo de herramienta, forma, diámetro, longitud, etc.

- Definir las fijaciones utilizadas para sostener el material durante las operaciones de mecanizado, para que no exista ningún problema que interfiera con la herramienta durante su trayectoria.

- Establecer las configuraciones para las operaciones de mecanizado, como las cajas delimitadoras del material, velocidad del husillo (spindle speed), velocidad de avance de las herramientas (feed rate).

- Crear las operaciones de manufactura, estas generaran las trayectorias que seguirán las herramientas (toolpath), mientras van removiendo material hasta completar la elaboración de la pieza. Las operaciones comunes incluyen las fases de desbaste, contorneado y acabado.

- Una vez generados los toolpaths para cada operación, estos pueden ser visualizados, y si no requieren ningún ajuste pueden ser exportados como código-G para ser utilizado por el Sistema CNC.

2.2.3 Herramienta PyCAM

PyCAM es una herramienta gratuita que permite la generación de toolpaths y su exportación como código-G para su uso en máquinas CNC de 3 ejes. PyCAM permite utilizar archivos CAD 3D de formato STL y diseños 2D en formatos DXF o SVG, además trabaja en Sistemas Operativos Linux, Windows y MacOS [12].

Las características más importantes de PyCAM incluyen:

Transformaciones del modelo: Rotar, voltear y reflejar un modelo alrededor de los ejes X, Y, Z, escalado y cambio de posición.

Configuración de Herramientas: Definición de una base de datos de herramientas con información como: Forma (cilíndrica, esférica, toroide), diámetro, Feedrate, Spindle speed. Además permite exportar un archivo con estos datos para su uso en LinuxCNC.

Configuración de Procesos: Definición de las estrategias que se usaran para generar la trayectoria en las diferentes etapas de tallado:

- Slice Removal: Para la fase inicial o desbaste, trata de remover la mayor cantidad de material cortando en capas de cierta altura.
- Contour: Traza una trayectoria de acuerdo al contorno del modelo.
- Surface: Para la etapa final de acabado, un tallado fino alrededor de toda la superficie.
- Engraving: Para el grabado o tallado de modelos 2D.

Configuración de Limites: Creación de cajas delimitadoras para especificar las áreas donde operaran las herramientas.

Lista de Tareas: Definición de las etapas del procesado del material como desbaste, semi-acabado, acabado, aquí se especifican las herramientas a utilizar, el proceso y las cajas delimitadoras, luego se genera una trayectoria (toolpath), su pre visualización y simulación.

2.3 LinuxCNC

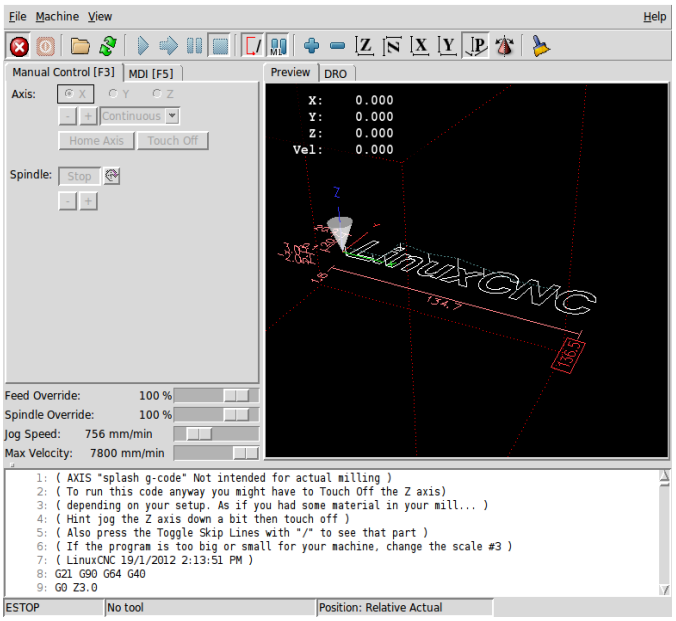


Figura 2.6 *Pantalla inicial LinuxCNC*

2.3.1 Descripción General

LinuxCNC (ver Figura 2.6) es un Software de Control CNC para máquinas-herramientas como fresadoras, tornos y enrutadores. LinuxCNC es un software gratuito de código abierto, soportado por una gran comunidad de desarrolladores. [13]

El manejo de una máquina CNC requiere, de un control preciso y en tiempo real del movimiento y otras tareas, por ello LinuxCNC requiere una plataforma con capacidad de cómputo en tiempo real. Para esto LinuxCNC utiliza un kernel de Linux con la extensión RTAI (Real Time Application Interface).

La página oficial de LinuxCNC provee de discos de instalación basados en la distribución Ubuntu con la extensión RTAI instalada y con LinuxCNC pre-compilado para facilidad de sus usuarios.

Entre sus características tenemos:

- Variedad de interfaces graficas Disponibles.
- Interprete de código-G
- No dispone de herramientas CAD/CAM para diseños ni generación de código-G.
- Manejos de máquinas de hasta 9 ejes
- Operación y retroalimentación de servo motor y motores de paso.
- Software PLC programable con diagramas de escalera.
- Fácil creación de configuraciones para máquinas-herramientas
- Sistema de planificación de movimiento en tiempo real
- Control del radio de corte y compensación de longitud de herramientas, movimiento sincronizado de ejes, feedrate adaptativo, control de velocidad constante.

LinuxCNC es capaz de controlar una gran variedad de maquinaria de diferentes modelos, solo es necesario crear la

configuración necesaria para la máquina y está listo para trabajar. LinuxCNC cuenta con Asistentes de Configuración como StepConf para máquinas CNC que utilicen motores de paso y sean controladas vía puerto paralelo, y PNCconf para tarjetas de control especializadas y también uso de servo motores. Además de esto LinuxCNC incluye configuraciones pre-establecidas de modelos de máquinas específicos.

La creación de la configuración de la máquina es facilitada por estos asistentes, en el capítulo 3 se detalla este proceso, y podemos resumir los parámetros más importantes a continuación:

Información General de la Máquina

- Tipo de configuración de los ejes: XYZ
- Unidades de medida: mm, pulgadas
- Información de los controladores de los motores
- Dirección del puerto paralelo

Configuración de Salidas del Puerto Paralelo

- Señales de Dirección y de Paso de los motores
- Señales de control para el husillo de la herramienta

Configuración de los ejes

- Información del motor: Pasos/revolución, micropasos, velocidad y aceleración máxima
- Dimensiones y límites del área de trabajo

2.3.2 Arquitectura

Hay 4 componentes principales en la arquitectura de LinuxCNC (ver Figura 2.7):

- Un controlador de movimiento (EMCMOT)
- Un controlador de entradas y salidas discreto (EMCIO)
- Un Controlador de tareas (EMCTASK)
- Múltiples Interfaces de Usuario (GUI)

Además de esto se encuentra la Capa de Abstracción de Hardware que permite configurar LinuxCNC sin necesidad de recompilar.

De los 4 componentes de LinuxCNC, solo el controlador de movimiento EMCMOT trabaja en tiempo real. La comunicación entre componentes que no son de tiempo real se realiza con NML y la comunicación con EMCMOT se realiza mediante un búfer de memoria compartida. [13]

Controlador de Movimiento (Motion Controller) EMCMOT

Este controlador está escrito en C para mayor portabilidad para sistemas operativos en tiempo real, este recibe los comandos desde el espacio de usuario a través de un búfer de memoria compartida para luego ejecutarlos en tiempo real. Este búfer también almacena la información del estado del controlador. Para controlar el movimiento de los motores de cada eje y otros dispositivos de hardware, el controlador hace uso de la capa HAL.

Entre los controles que realiza están el muestreo de las posiciones de los ejes, cálculo del punto siguiente en una trayectoria, interpolación entre puntos, y el cálculo de la salida para los motores en forma de señales, con una metodología propia para los servo-motores y motores de paso. El controlador también utiliza archivos de inicialización .INI para configurar parámetros del sistema como número y tipo de ejes, sus unidades de medida, factores de escala, entre otras. [14]

Controlador de I/O Discretas (EMCIO)

Está escrito en C++ usando la librería NIST RCS Library (Real Time Control Systems Library) y consiste en una jerarquía de

clases en C++ de controladores que se comunican mediante NML (Neutral Message Language) el cual es un API para funciones de comunicación que soporta varios protocolos comunes. Estos controladores son bastante específicos de la máquina y no son configurables como los controladores de movimiento, entre estos están los sistemas de refrigerante, lubricante, el controlador de la herramienta [14].

Controlador de Tareas (EMCTASK)

Este controlador se encarga de monitorear y coordinar a los controladores EMCMOT y EMCIO, recibiendo los comandos de la interfaz de usuario (GUI), los analiza basado en el estado actual de los demás controladores y llama a funciones necesarias para su ejecución o envía comandos a los propios controladores. Este controlador también se encarga de interpretar el código-G en el Sistema [13].

Interfaz de Usuario (GUI)

LinuxCNC viene con una gran variedad de Interfaces como AXIS la cual cuenta con una pre-visualización del modelo a elaborar, NGCGUI especializada en subrutinas, Touchy para paneles táctiles, entre otras. Estas interfaces se comunican con el

controlador de Tareas EMCTASK utilizando NML, envía mensajes como encendido/apagado, controles de ejecución del código, pausa, detener, movimiento manual de ejes, etc. [14]

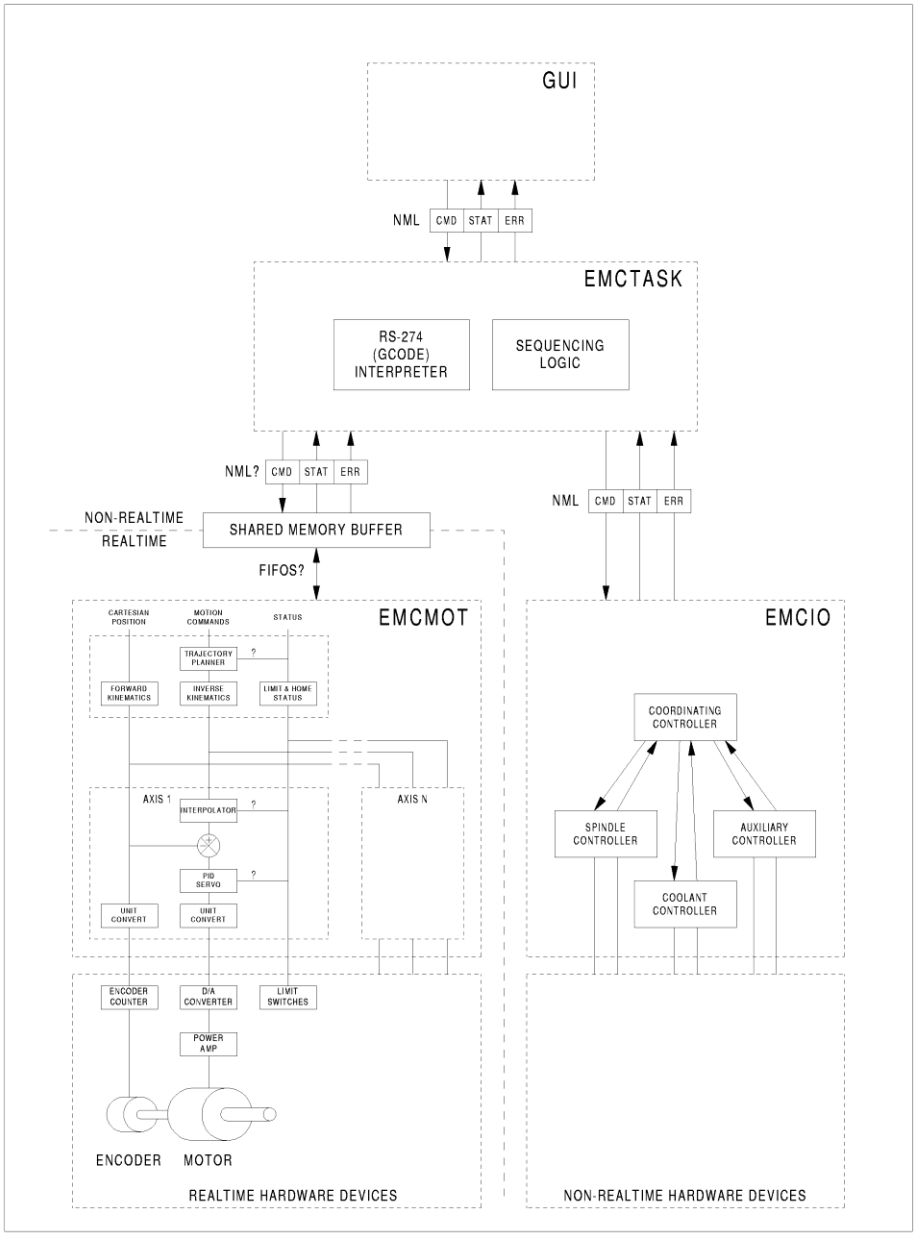


Figura 2.7 *Arquitectura de LinuxCNC. [13]*

2.3.3 Capa de Abstracción de Hardware (HAL)

La capa de Abstracción de Hardware es medio o una interfaz que permite al software acceder a recursos de hardware fácilmente, en LinuxCNC fue diseñado para configurar dispositivos de hardware de una manera más fácil, comunicándose con una representación abstracta de esa componente a través de un controlador.

HAL está basado en los mismos principios usados para diseñar sistemas de hardware o circuitos, por ejemplo un sistema está formado por componentes interconectados, para ello un componente debe poseer terminales y para su conexión con otro componente necesita un medio como un cableado, todos estos elementos tendrán una representación en software lo cual describiremos a continuación.

Componentes

En una máquina CNC un componente sería un controlador de pasos, motores, sensores, etc. Su equivalente en HAL sería un fragmento de software con entradas, salidas y comportamiento definido.

Parámetro

Varios componentes tienen ajustes que no están conectados a otros componentes pero que su información es requerida. En HAL son llamados parámetros y hay 2 tipos, parámetros de entrada definidos por el usuario y parámetros de Salida.

Pin

En Hardware serían los terminales que permiten conectar los componentes unos a otros, en HAL son llamados pines, donde cada pin tiene su nombre requerido para la conexión.

Señal

Su equivalente en hardware serían los cables que conectan los pines o terminales.

Tipos

Ambos pines y señales tienen tipos, las señales solo pueden ser conectadas a pines del mismo tipo y son los siguientes:

- bit – TRUE/FALSE or ON/OFF value
- float - a 64 bit floating point value,
- u32 - a 32 bit unsigned integer
- s32 - a 32 bit signed integer

Funciones

Los componentes de hardware operan inmediatamente al recibir sus entradas, sin embargo los componentes HAL requieren el

uso de funciones para realizar su trabajo. Un ejemplo funciones comunes serían las de lectura y escritura.

Un ejemplo de una componente HAL podría ser el puerto paralelo como se muestra en la Figura 2.8, en él se definen hasta 3 pines HAL para cada uno de los 17 pines físicos utilizables, uno para representarlo como pin de salida, otro como entrada y el último como entrada invertida. Esta componente necesita por lo menos 2 funciones una de lectura que lea los pines físicos y actualice el pin HAL y otra que escriba la información de HAL al pin físico, ambas implementadas en el código del controlador.

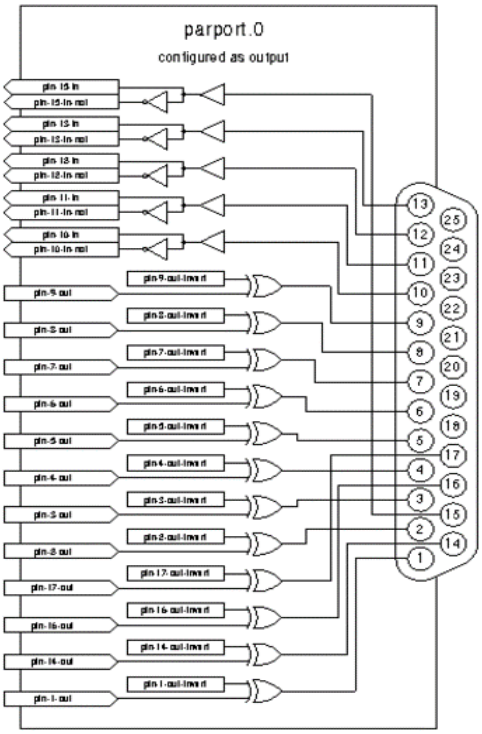


Figura 2.8 *Componente HAL Parport. [13]*

CAPÍTULO 3

3. REQUERIMIENTOS Y ARQUITECTURA DEL SISTEMA

3.1 Requerimientos del Sistema

Los requerimientos que cumplirá el Sistema se han dividido en funcionales y no funcionales.

Requerimientos funcionales

El Sistema deberá:

- Controlar una máquina-herramienta de 3 ejes independientes.
- Recibir un archivo de código G codificado en ASCII de cualquier herramienta CAM comercial o software libre.
- Permitir el cambio manual de herramienta durante la ejecución de los procesos de tallado.
- Tener la opción de conexión de interruptores y sensores de fines de carrera.

Requerimientos no funcionales

- Poseer una interfaz amigable al usuario y compatible con pantalla táctil.
- Proporcionar tiempos de respuesta aceptables, con baja latencia en el orden de los nanosegundos.
- Garantizar una velocidad estable de funcionamiento y precisión en el movimiento de los motores de la máquina.

3.2 Visión general del Sistema

La arquitectura del Sistema de Control CNC se encuentra distribuida en varios componentes como se muestra en la Figura 3.1.

El computador Industrial, basado en una plataforma embebida, el cual incluye las herramientas de Software CAD/CAM y el Software de Control que mediante el puerto paralelo envía las señales a la Tarjeta Interfaz, esta distribuye las señales hacia cada uno de los controladores de los motores de paso para el movimiento de la herramienta de la maquina CNC, también tiene la capacidad de recibir señales de sensores y switches para transferirlos al software de control.

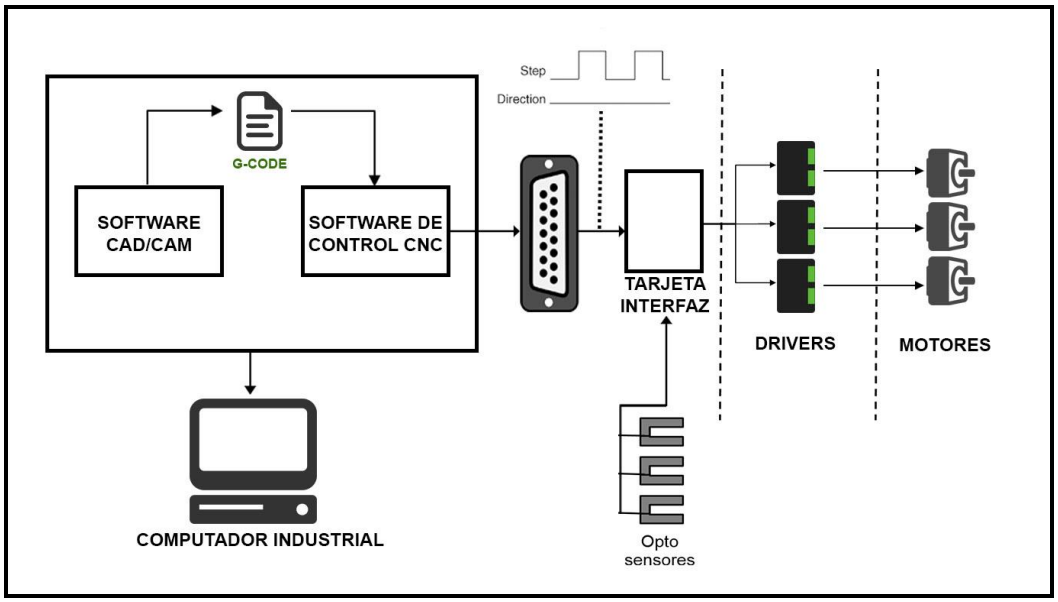


Figura 3.1 *Arquitectura del Sistema – Componentes*

3.3 Componentes del Sistema

3.3.1 Sistema de Control CNC

3.3.1.1 Características Generales

En este proyecto se utiliza un Sistema de Control CNC basado en PC, debido a que su uso se ha extendido en la actualidad y poseen la ventaja de poder incorporar todo el software necesario para el proceso de manufactura en la misma máquina, además de que permite realizar modificaciones con mayor facilidad comparado con los sistemas de arquitectura tradicional.

Se usó una plataforma embebida para obtener un Sistema compacto de nivel Industrial, en este proyecto se utilizó el

embebido Advantech PCM-9562 por que cumple con los requerimientos básicos para operar máquinas CNC, principalmente por contar con un puerto paralelo el cual es el medio básico para el envío de señales a los motores de la máquina, además se realizaron pruebas con un embebido adicional, VIA EPIA-P830 con la finalidad de experimentar la utilización de los pines de propósito general GPIO como medio de conexión alternativa con la máquina CNC.

Entre las características básicas que se requiere para operar, se detallan las siguientes:

Plataforma: LinuxCNC funciona en una plataforma de 32 bits, el uso de una plataforma de 64 bits se encuentra en desarrollo y requiere una instalación personalizada.

Procesador: Compatible con la arquitectura x86, 700MHz o mejor solo para Ubuntu y LinuxCNC.

RAM: El mínimo recomendado solo para la distribución de Ubuntu y LinuxCNC es 512 MB.

Entradas y Salidas: Monitor o Pantalla Táctil, Teclado, Mouse, Puerto Paralelo.

3.3.1.2 Embebido Advantech PCM-9562

Figura 3.2 *Embebido Advantech Pcm-9562. [15]*

Se escogió esta tarjeta embebida para este proyecto ya que cuenta con la conexión de un puerto paralelo (ver Figura 3.2), el cual es de suma importancia ya que es el medio de comunicación para el Sistema CNC. Además de que posee otras características de importancia como ranuras de expansión PCI, puertos seriales útiles para el uso de una pantalla táctil. A continuación se detallan en la Tabla I algunas de las especificaciones más importantes, el datasheet completo se puede encontrar en el anexo A. [15]

Tabla I.- *Especificaciones Embebido Advantech PCM-9562. [15]*

CPU	Intel Atom N450/D510 1.66 GHz
Factor de Forma	EBX
Dimensiones	203 x 146 mm
Memoria	DDR2 667/800MHz de hasta 2GB
Almacenamiento	SATA II x3, CompactFlash tipo I/II
I/O	Serial 4 x RS-232, 2 x RS-422/485 Ethernet RJ45 PS2/KB, Mouse USB 2.0 x8 Paralelo/LPT GPIO 16-bit VGAx1, LVDSx1
Expansion	PCI, mini PCIe, PC/104-Plus

3.3.1.3 Embebido VIA EPIA-P830 Pico-ITX

Este embebido se lo utilizo como medio de prueba para demostrar la posibilidad de utilizar pines GPIO en reemplazo del puerto paralelo (ver Figura 3.3).

Figura 3.3 Embebido VIA EPIA-P830. [16]

Las características más importantes se detallan en la Tabla II, el datasheet completo se puede encontrar en el anexo B. [16]

Tabla II.- *Especificaciones Embebido EPIA-P830. [16]*

CPU	1.2GHz VIA Nano® E-Series
Factor de Forma	Pico-ITX
Dimensiones	10 cm x 7.2 cm
Memoria	DDR3 800/1066 SODIMM socket de hasta 4GB
Almacenamiento	SATA x2
I/O	Gigabit Ethernet PS2/KB, Mouse USB 2.0 x5 + 2 back panel 1 DIO pin connector (4GPI+4GPO) HDMIx1, VGAx1, LVDSx1 12V DC-in power connector

3.3.1.4 Software CAM

Se utilizara PyCAM, la cual es una herramienta de software libre y multiplataforma usada para la creación de la trayectoria de la herramienta y su representación en código G para máquinas de 2 a 3 ejes (ver sección 2.2.3).

3.3.1.5 Software de Control CNC

El software utilizado será LinuxCNC, un Sistema de Control de Máquinas CNC mediante código G, se lo escogió por su amplia gama de características que nos permiten cumplir nuestros

requerimientos. Requiere de un kernel de Linux modificado con una extensión de tiempo real para la generación de pulsos mediante software sin la necesidad de hardware adicional (ver sección 2.3).

3.3.2 Controlador de Motor de Paso

Se utilizaron controladores genéricos para el manejo de los motores de la impresora 3D, estos controlan los motores recibiendo las señales de LinuxCNC a través del puerto paralelo y convirtiéndolas en movimiento.

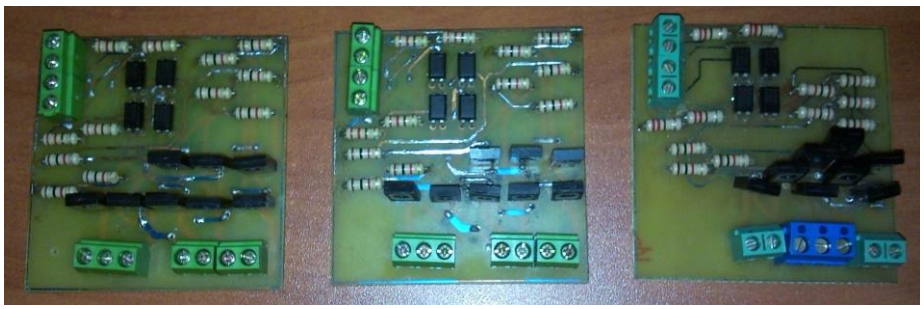


Figura 3.4 *Controladores de motor de paso*

3.3.3 Impresora 3D: Prusa 3D Printer Kit

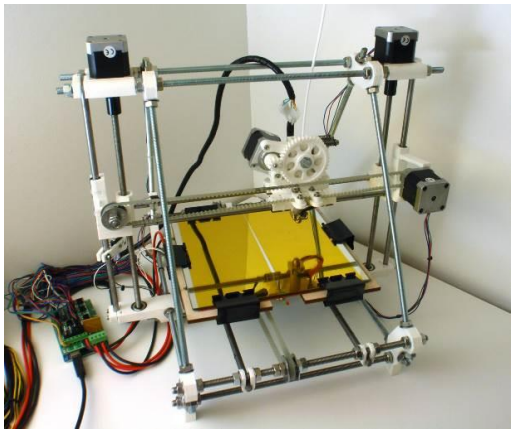


Figura 3.5 *Impresora 3D Prusa Mendal. [17]*

La Figura 3.5 muestra una impresora 3D modelo Prusa Mendel, se utilizó uno de estos modelos como prototipo de prueba de una máquina CNC ya que tienen en común el uso de motores de paso y de hecho suelen trabajar con código-G para crear sus modelos.

La impresora Prusa Mendel comprende la 2da generación de impresoras 3D, su área de trabajo tiene un volumen de 200mm x 200mm x 100mm, la impresora incluye 5 motores de paso, dos para el eje Z, uno para el eje X, uno para el eje Y, y uno para el extrusor, también posee opto sensores para controlar los límites de los ejes. Son populares por su relativo bajo costo y su facilidad de ensamblaje. [17]

CAPÍTULO 4

4. IMPLEMENTACIÓN DEL SISTEMA

4.1 Implementación del Sistema en el Embebido

4.1.1 Instalación de LinuxCNC

La instalación se realizó utilizando el Live-CD obtenido en la página oficial de LinuxCNC. Esto brinda la ventaja de tener una distribución de Ubuntu automáticamente configurada y lista para usar LinuxCNC. La distribución usada fue Ubuntu 10.04 Lucid Lynx/LinuxCNC.

Para poder desarrollar el controlador de GPIO es necesario instalar LinuxCNC desde el código fuente, para eso utilizamos los programas “apt-get” y “git” desde el Terminal de Linux.

Primero instalamos algunos paquetes necesarios para compilar LinuxCNC utilizando apt-get:

```
sudo apt-get install libpth-dev
sudo apt-get build-dep linuxcnc
```

El código fuente de LinuxCNC está almacenado en Git, por lo que necesitamos instalarlo:

```
sudo apt-get install git-core gitk git-gui
```

Luego creamos una copia del código fuente del proyecto en una carpeta a la que llamaremos “linuxcnc-dev” dentro del directorio local de Ubuntu.

```
git clone git://git.linuxcnc.org/git/linuxcnc.git
linuxcnc-dev
```

De esta manera obtendremos la versión más reciente del código de LinuxCNC. Para obtener una versión anterior o alguna específica podemos usar:

```
cd linuxcnc-dev
git checkout v2.5_branch
```

o usar:

```
git checkout v2.5.0
```

Para asegurarnos que no nos falte ninguna dependencia utilizamos el siguiente código:

```
cd debian
./configure -a
```

```
cd ..
dpkg-checkbuilddeps
```

Luego utilizamos sudo apt-get install para instalar todas las dependencias faltantes.

Al instalar la última versión (2.6) también necesitamos instalar la siguiente dependencia:

```
sudo apt-get install libboost-python-dev
```

Para compilar LinuxCNC nos dirigimos al directorio creado “linuxcncn-dev” y ejecutamos los siguientes comandos:

```
cd src
./autogen.sh
./configure --enable-run-in-place
make
make install-menus
sudo make setuid
```

Para ejecutar LinuxCNC nos volvemos a dirigir al directorio “linuxcnc-dev” y utilizamos los siguientes comandos:

```
. ./scripts/rip-environment
linuxcnc
```

4.1.2 Creación de la Configuración de la Máquina

En esta sección se mostrarán los pasos para obtener el archivo de configuración de nuestra máquina CNC, en este caso la impresora 3D Prusa Mendel.

Test de Latencia

La latencia es el tiempo que tarda el PC en detener sus actividades y responder a una solicitud externa. Cuanto menor sea la latencia, más rápidos y suaves serán nuestros pulsos de paso.

Podemos iniciar la prueba desde el menú “Aplicaciones>CNC>Latency Test” o desde la pantalla inicial del asistente de configuración Stepconf**.**

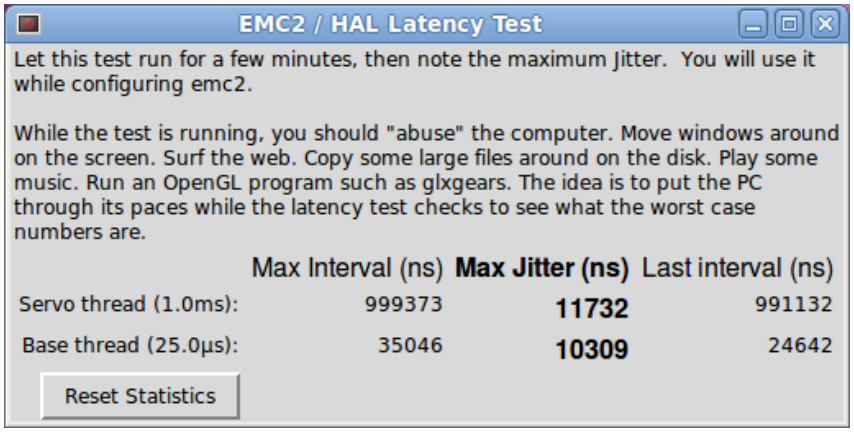


Figura 4.1 *Test de Latencia*

Mientras el test se ejecuta como se muestra en la Figura 4.1, se debe someter a la PC a una prueba de estrés, dándole una intensa carga de trabajo, mover ventanas, copiar grandes cantidades de archivos en el disco, reproducir música, videos, software de manejo de gráficos, OpenGL, etc., debemos ejecutar

el test el mayor tiempo posible, hasta que se encuentren los valores del peor escenario.

Los valores importantes son los de “Max Jitter”**,** escogemos el mayor y lo guardamos para utilizarlo más adelante durante la creación del archivo de configuración. Si nuestro valor es menor que 15,000 o 20,000 nanosegundos nuestra computadora debería dar muy buenos resultados con LinuxCNC, entre 30,000 – 50,000 ns también tendría buenos resultados, si el valor es mayor a 100,000 ns nuestra computadora no sería adecuada para trabajar con LinuxCNC.

En nuestro caso obtuvimos un valor máximo de 11732 ns que es aceptable dentro de lo recomendado para operar.

Asistente de configuración Stepconf

Stepconf es un programa que permite la creación de los archivos de configuración para LinuxCNC para máquinas controladas vía puerto paralelo y que utilizan señales de paso y dirección. El asistente Stepconf viene instalado junto a LinuxCNC y se encuentra en el menú Aplicaciones>CNC>LinuxCNC Stepconf Wizard.

Al iniciar el asistente se nos muestra opciones para crear una nueva configuración o modificar una existente. Además nos permite crear accesos directos, uno hacia la carpeta con los archivos y otro para iniciar LinuxCNC con esta configuración automáticamente.

Información Básica de la Máquina

Se nos mostrará una ventana como en la Figura 4.2, empezamos nombrando a nuestra máquina "impresora3D", la configuración de tres ejes XYZ y como unidad milímetros. A continuación se necesita ingresar la información del controlador de los motores, en este caso los controladores fueron diseñados en el Centro de Visión y Robótica de la ESPOL, por lo que utilizaremos los valores predeterminados, para un controlador comercial es necesario consultar estos valores en su ficha técnica, para maximizar su eficiencia:

- Step Time: El tiempo en nanosegundos que dura el pulso de paso.
- Step Space: El tiempo mínimo entre pulsos, en nanosegundos.
- Direction Hold: Cuánto tiempo el pin de dirección se mantiene después de un cambio de dirección.
- Direction Setup: El tiempo que se mantiene el estado del pin de

dirección antes de que cambie, después de un pulso de paso.

Lo siguiente es definir la dirección base del puerto paralelo, cualquier PC que cuente con uno tendrá la dirección 0x378, también se pueden especificar hasta dos puertos paralelos extra, el siguiente dato es el valor de Jitter máximo que obtuvimos en el Test de Latencia, y finalmente tenemos la opción del cambio manual de herramienta con lo que el sistema realizará una pausa y pedirá nuestra confirmación para continuar al momento que sea necesario cambiar nuestra herramienta durante la operación.

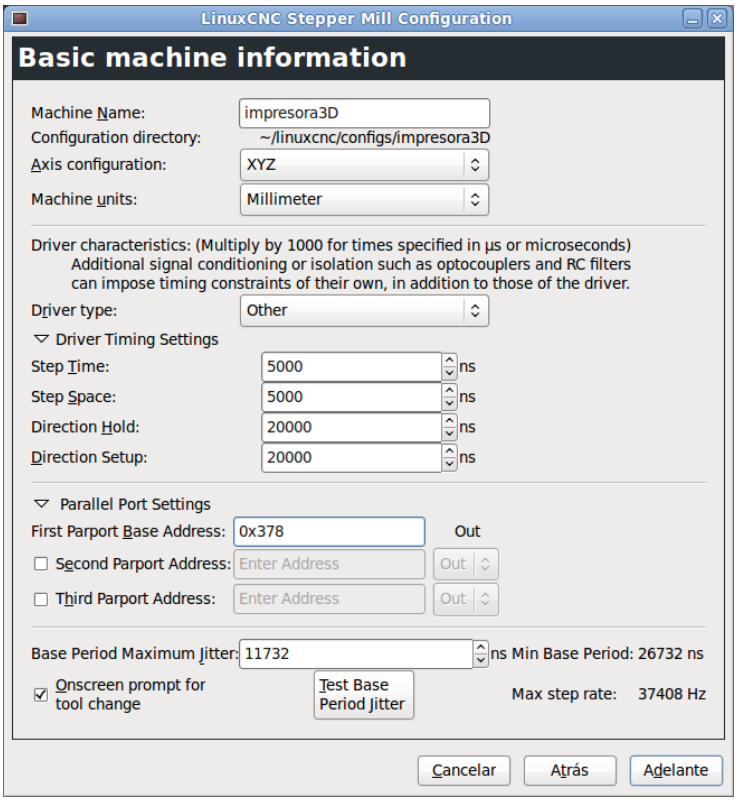


Figura 4.2 *Informacion Basica de la Máquina*

Configuración del Puerto Paralelo

La siguiente ventana será la mostrada en la Figura 4.3, aquí seleccionaremos las señales de salida (dirección y paso de los motores) y entrada (botones, sensores, switches para límites y proceso de homing) para los pines del puerto paralelo. También se pueden elegir algunos ajustes predefinidos para máquinas Sherline y Xylotex.

Seleccionamos la opción invertir en caso de que la lógica de la señal necesitada sea invertida. Si alguno de los pines no es utilizado lo dejamos como Unused.

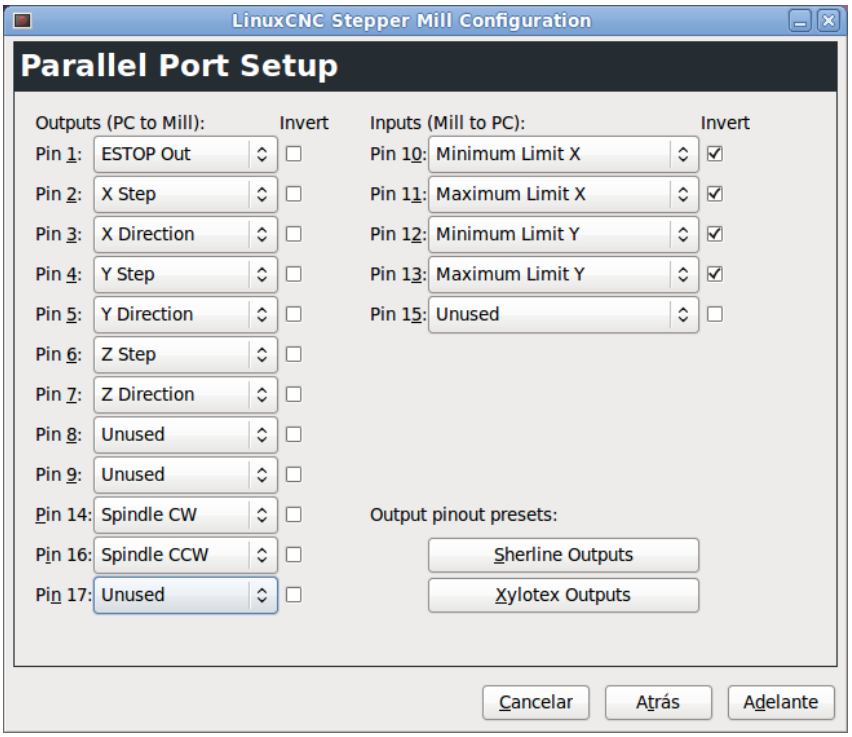


Figura 4.3 *Configuración de las salidas del Puerto Paralelo*

Configuración de ejes

Aquí indicamos información concerniente a como se moverá el motor de cada eje (ver Figura 4.4). Los parámetros son los siguientes:

• **Motor Steps Per Revolution** – El número de pasos por revolución del motor, por lo general son 200 pasos por revolución.

• **Driver Microstepping** – La cantidad de microstepping soportada de acuerdo a la configuración del controlador del motor de paso, si no se utiliza se deja un valor de 1.

• **Pulley Ratio** – Si la máquina utiliza poleas de diferente tamaño ingresar la relación del radio, caso contrario dejarlo como 1:1.

• **Leadscrew Pitch** – En caso de que las unidades sean pulgadas, ingresar el número de revoluciones del motor por pulgada, si es en milímetros el número de milímetros por revolución, en nuestro la configuración fue la siguiente:

- **Eje X Y**: 50mm/rev **Eje Z**: 1.4mm/rev

• **Velocidad Máxima y Aceleración Máxima**, ambos valores se determinan mediante experimentación ver sección 5.2.

• **Home Location** – Posición home, es una posición predeterminada respecto al origen de coordenadas de la máquina, se la utiliza como punto de partida al empezar a operar,

la máquina se dirige a esa posición de forma automática al finalizar una secuencia de homing mediante el uso de interruptores o sensores o también se la puede definir de forma manual moviendo la herramienta de la máquina.

• **Table Trave**l – Los límites en los cuales el eje se puede mover, basado en el origen de coordenadas de la máquina (Ej.: (-100 a 100) (0 a 200) [mm o pulgadas]). En nuestra impresora 3D el área de trabajo es de 200x200x100 en el eje XYZ respectivamente.

• **Home Switch Location** – La posición del sensor o interruptor para la secuencia de homing. Si el sensor también es usado como final de carrera es recomendable que no se encuentre en la misma posición home definida previamente.

• **Home Search velocity** – La velocidad de la fase inicial de la secuencia de homing en búsqueda del sensor, el signo de la velocidad define la dirección en la que se buscara el sensor.

• **Home Latch direction** – La dirección en la que se realizará la fase final de homing después de activar el sensor, si escogemos “igual”, la maquina retrocederá un poco y volverá a avanzar hacia el sensor lentamente hasta activarlo de nuevo, si escogemos “opuesta” la maquina retrocederá lentamente hasta desactivarlo.

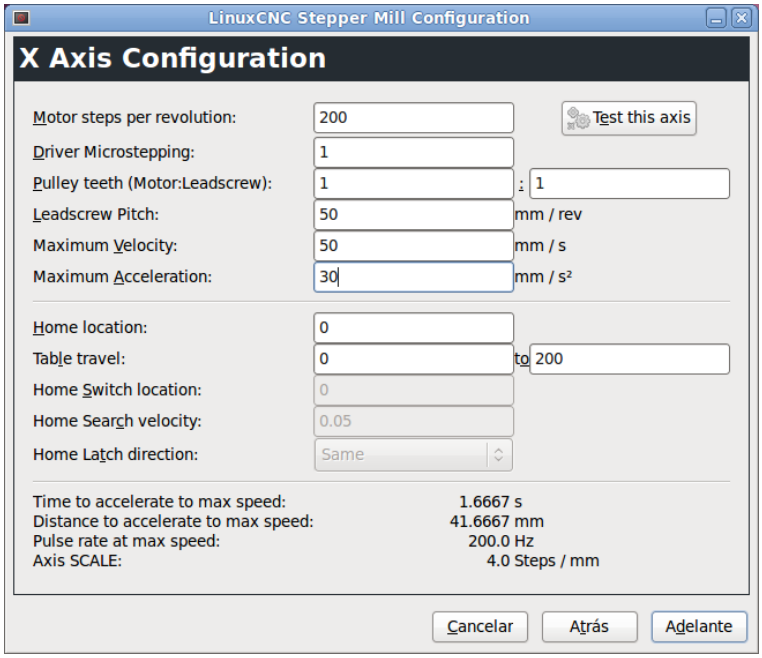


Figura 4.4 *Configuracion de ejes*

De esta forma se completa la creación de la configuración creándose la carpeta con el archivo de inicialización “impresora3D.ini” el cual se puede observar en el anexo C y también el archivo HAL “impresora3D.hal” con la información de la máquina.

4.2 Generación de la Trayectoria de la Herramienta

4.2.1 Modelo 3D

PyCAM utiliza archivos STL en formato ASCII, para nuestra demostración usaremos el modelo de un pedal (ver Figura 4.5).

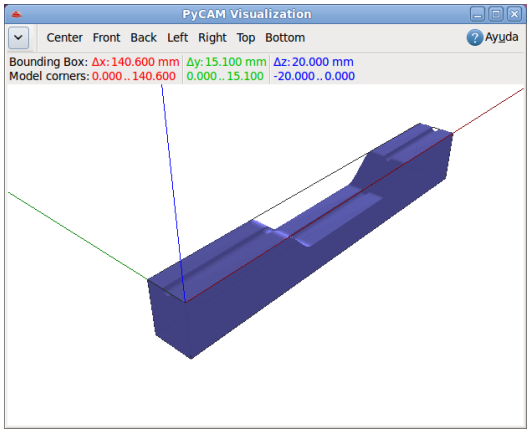


Figura 4.5 *Modelo CAD de un pedal*

4.2.2 Definición de las Características de las Herramientas

En la pestaña Tools se definirán las herramientas que se utilizarán durante todo el proceso. Es común utilizar herramientas grandes para las operaciones más básicas como remover material y herramientas más pequeñas para las operaciones de acabado. Las herramientas que definiremos serán: Cilíndrica (5mm) y una herramienta esférica (3mm) como se muestra en la Figura 4.6.

Los valores de Feedrate se refieren a la velocidad de desplazamiento de la herramienta y Spindle Speed a la de rotación, estos valores son dependientes del material en que se trabaje, sea madera, metal, plástico, y las características de la herramienta, son valores determinados mediante fórmulas

matemáticas o por experiencia del operario, en nuestro caso usaremos valores de Feedrate (500mm/min) y Spindle Speed (10000RPM).

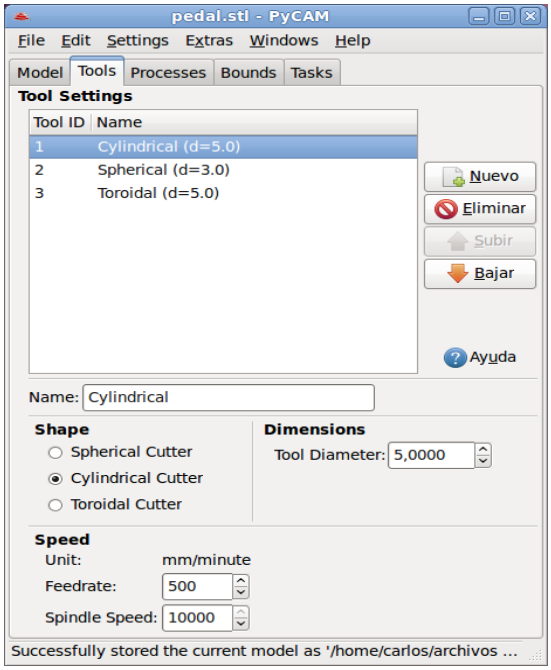


Figura 4.6 *Definición de Herramientas*

4.2.3 Definición de los Procesos

En la pestaña **Processes** se definirán las diferentes estrategias que se usaran en la generación de las trayectorias de las herramientas (**Toolpath**). Las estrategias que utilizaremos se muestran en la **Tabla III**:

Tabla III.- *Definición de Procesos de Tallado*

Proceso	Estrategia	Overlap [%]	Material Allowance	Max Step Down
Remover Material	Slice Removal	0	0,50	3 mm
Acabado	Surface	60	0	-

Utilizaremos solo 2 procesos, uno para remover la mayor cantidad de material posible y otra para la fase de acabado. El parámetro “Overlap” indica el porcentaje de distancia en la que se superpondrán las líneas de trayectoria, esto permite un tallado más preciso al aumentar el número de líneas y por eso se lo utiliza en el acabado, “Material allowance” define la distancia extra que se dejara de material, por ejemplo 0.5mm, se lo utiliza en las primeras etapas de tallado y cuando se utilizan herramientas no muy precisas, en la etapa de acabado este valor debe ser cero.

Max. Step Down es la altura máxima de las capas en las que se divide el modelo para la operación de tallado. Este valor depende de la dureza de material, y las características de la herramienta utilizada.

También tenemos otras opciones como “Path Direction” donde se especifica si las líneas del toolpath se deben mover a lo largo del eje X, Y o ambos, y también el estilo de fresado, sea convencional o climb.

4.2.4 Generación de Código G

PyCAM utiliza tareas para detallar los ajustes del toolpath a ser generado, básicamente son la combinación de las herramientas, procesos y las cajas delimitadoras que debemos tener definido previamente. Las que utilizaremos se detallan en la Tabla IV.

Tabla IV.- *Definición de Tareas de Tallado*

Tarea	Herramienta	Proceso	Limites
Desbaste	Cilíndrica (5mm)	Remover Material	Margen 10%
Acabado	Esférica (3mm)	Acabado	1mm

Una vez hecho los ajustes podemos generar los toolpaths con el botón “Generate Toolpath” manteniendo una tarea seleccionada, o mediante el botón “Generate All” que generara toolpaths de todas las tareas marcadas en la lista.

Los toolpaths generados aparecerán listados en la nueva pestaña Toolpaths**,** además en la ventana de visualización se podrá observar a cada uno de ellos como se muestra en la Figura 4.7. A través del botón Simulate podremos observar la trayectoria que seguiría la máquina con el toolpath seleccionado.

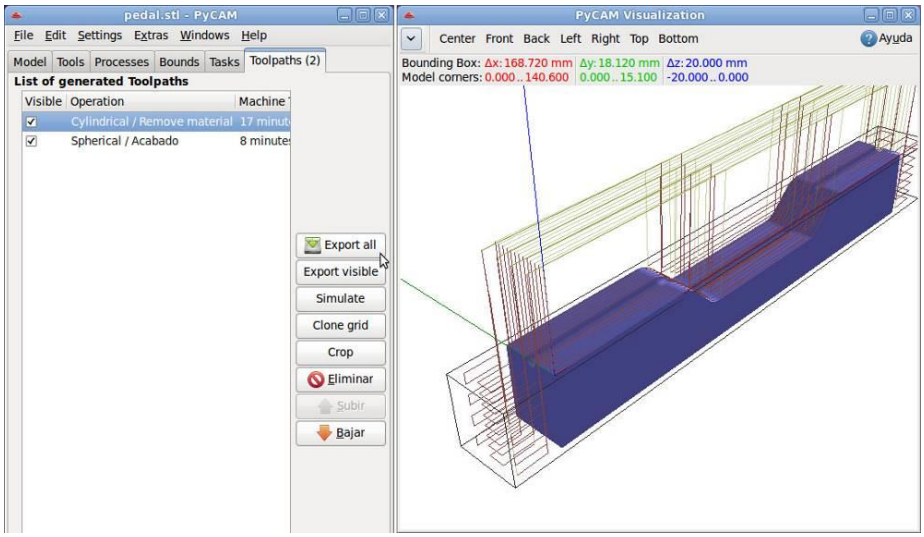


Figura 4.7 *Toolpaths y su Simulacion en PyCAM*

Una vez que confirmemos que todo está correcto, nos aseguramos de que los toolpaths estén en el orden adecuado y presionamos el botón Export all lo que nos permitirá guardar el G-Code en un archivo .ngc.

4.3 Diseño del Controlador GPIO para la Capa de Abstracción de Hardware

En esta sección se explicara el diseño de un Controlador para el uso de pines GPIO en el embebido VIA EPIA-P830 como alternativa extra al uso del puerto paralelo. El proceso en general consiste en primero conocer las direcciones de memoria de los pines GPIO además de las funciones necesarias para poder leer y escribir datos en ellos, las cuales las podemos encontrar en la documentación del embebido, luego se procede a crear un componente HAL en representación del puerto físico asociada a una estructura de datos que almacenen sus valores, y la creación de las funciones necesarias para lectura y escritura de información entre la componente HAL y los pines físicos.

4.3.1 Programación en GPIO

La localización de pines los pines GPIO en el embebido VIA EPIA-P830 se muestra en la Figura 4.8 y la Tabla V.

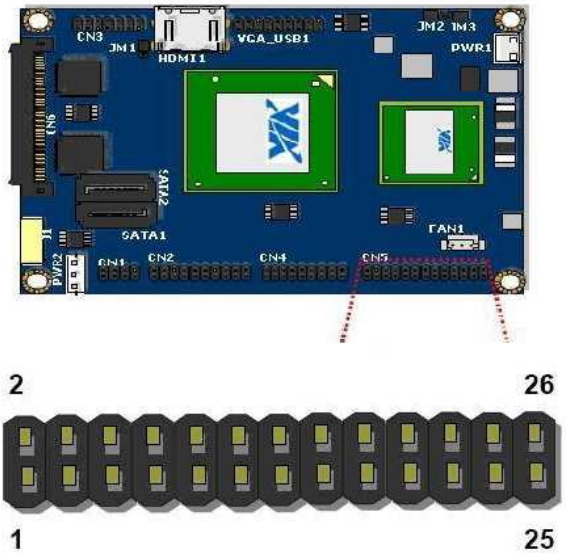


Figura 4.8 *Localizacion de los pines GPIO. [16]*

Tabla V.- *Distribución de pines GPIO. [16]*

Pin	Signal	Pin	Signal
17	GPIO12	18	GPI8
19	GPIO9	20	GPI9
21	GPO12	22	GPI5
23	GPIO1	24	GPI4
25	GND	26	GND

Tabla VI.- *Información de los pines GPIO. [16]*

Signal	Pin No.	Signal Select Register	Signal Description
GPI4	AN34		GPI4 input value on PMIO_48[4]
GPI5	AL32		GPI5 input value on PMIO_48[5]
GPI9	AB36		GPI9 input value on PMIO_49[1]
GPI8	AV35		GPI8 input value on PMIO_49[0]
GPIO1	AT33	PMU_RX95[3]=1 PMU_RX95[2]=1	GPIO1 output control on PMIO_4D[4]
GPO12	AT35	PMU_RX9B[0]=1	GPO12 output control on PMIO_4F[2]
GPIO9	AA31		GPIO9 output control on PMIO_4E[4]
GPIO12	Y34	PMU_RX80[6]=0	GPIO12 output control on PMIO_4E[7]

En la Tabla VI se muestra en la columna **"Signal Select Register"** los registros de memoria y el valor que necesitan ciertos bits para habilitar algunos de los pines, esta inicialización se requiere previo a utilizarlos. En la columna **"Signal Description"** se muestran los registros y los bits específicos donde se encuentran los valores de entrada y salida que debemos leer o escribir.

Según la Guía Operativa del Embebido el proceso para utilizar uno de los pines es el siguiente:

a) Obtener la dirección base de PMIO

```
#define VX900 0x80008800
WORD wPmioBase;

GetPMIOBaseAddr()
{
DWORD pciAddr;
/* For (VX900)  PMIO Base Address is located in
   PCI configuration space  Bus 0, device 17, function 0, offset 0x88
*/
pciAddr = VX900 | 0x88;
outpd(0x0CF8, pciAddr);
wPmioBase = inpd(0x0CFC);
wPmioBase &= 0xFFFE;
printf("PMIO Base address = %x\n", wPmioBase);
} /* end GetPMIOBaseAddr */
```

Necesitamos la dirección de PMIO porque desde ahí se puede acceder a los pines GPIO, para eso necesitamos la dirección de

memoria donde se encuentra dentro del espacio de configuración PCI y la almacenamos en **pciAddr**, y luego escribimos esta dirección en el puerto estandarizado 0xCF8 que corresponde al "*Configuration Space Address*", con este método conseguimos los datos almacenados en esa dirección leyéndolos a través del puerto 0xCFC o *"Configuration Space Data"*.

b) Habilitar el pin GPI/GPO. Por ejemplo el GPO12

```
GPOEnable()
{
DWORD pciAddr;
DWORD value;
/* For (VX900)
* GPO Control is located in PCI configuration space
* Bus 0, device 17, function 0, offset 0x9B     */
pciAddr = VX900 | 0x9B;
outpd(0x0CF8, pciAddr);
value = inpd(0x0CFC);
/* eg. Enable GPO 12
* GPO 12: Rx9B[0] = 1                    */
value |= 0x01000000; /* set 1 to bit 24 */
outpd(0x0CFC, value);
} /* end GPOEnable */
```

Para habilitar el pin **GPO12** se realiza un proceso similar al paso anterior, escribimos la dirección donde se encuentra el control de este pin en el puerto 0xCF8, y luego leemos el dato almacenado en el puerto 0xCFC, modificamos el bit requerido para habilitar el pin y reemplazamos los datos en la misma dirección 0xCFC.

c) Escribir en el pin GPO12

```
// Put byte to GPO (eg. Pull high GPO 12)
GPOSet()
{
BYTE data;

// GPO 12 is in the offset 0x4F bit 2
data = inp(wPmioBase + 0x4F);
data |= 0x04;

// Out the value to the GPO io address
outp(wPmioBase + 0x4F, data);
} /* end GPOSet */
```

Para escribir en el pin GPO12 utilizamos la dirección PMIO base más el offset correspondiente, leemos los datos y alteramos el bit adecuado, dándole un valor de 1, y lo reemplazamos en la misma dirección de memoria.

4.3.2 Creación y compilación del Controlador en LinuxCNC

A continuación se detallaran las partes más importantes del diseño del componente HAL, una versión completa del código se puede encontrar en el anexo D.

En la Figura 4.9 se muestra el diseño de la componente HAL, tenemos nuestra estructura de datos viagpio_t la cual está asociada a los pines de nuestra componente hal_viagpio que a

su vez representa a los pines físicos, tenemos también las funciones de lectura y escritura para interactuar con ellos.

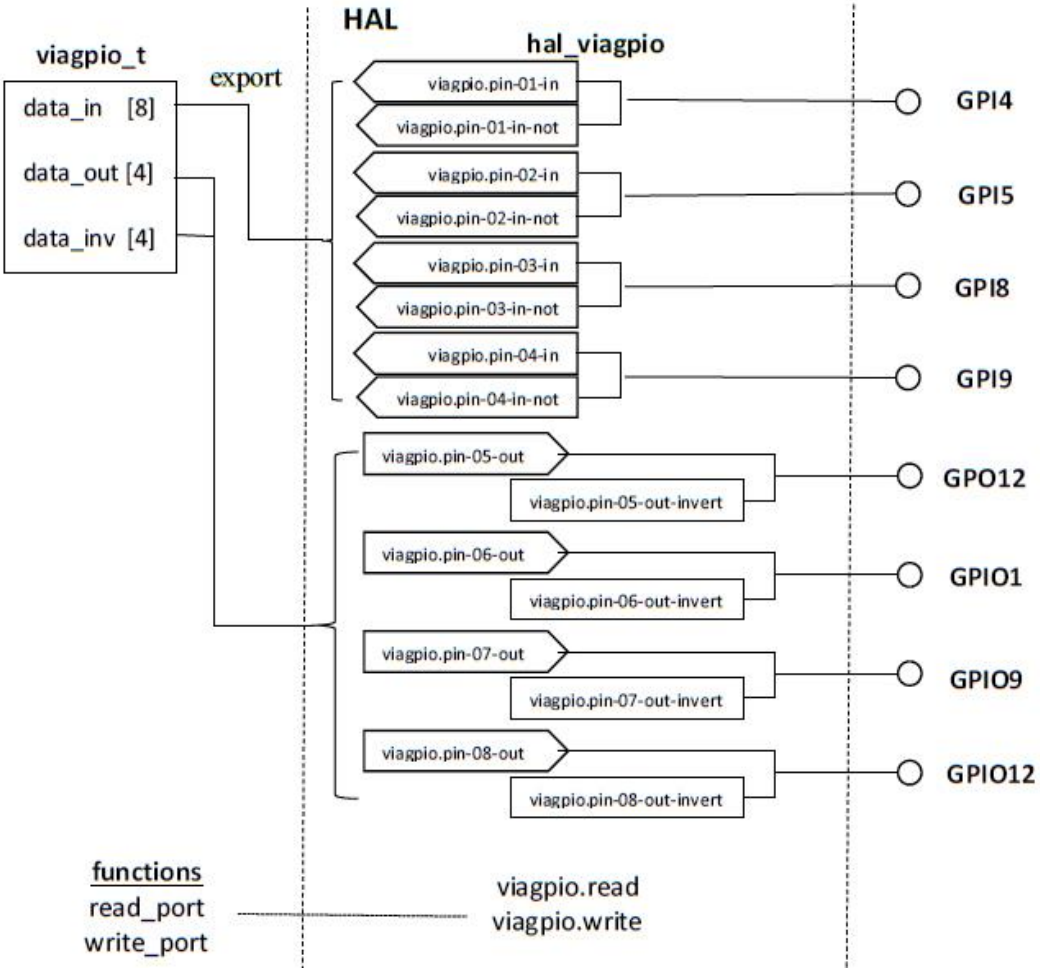


Figura 4.9 *Diseño de la componente hal_viagpio*

Librerías

Las librerías utilizadas incluyen varias del API realtime (RTAPI), la librería HAL, <asm/io.h> para el uso de las funciones de lectura y escritura en puertos, por ejemplo inl(), outl().

```
#include "rtapi.h"       /* RTAPI realtime OS API */
#include "rtapi_ctype.h" /* isspace() */
#include "rtapi_string.h"/*strcmp() */
#include "rtapi_math.h"
#include "rtapi_app.h"  /*RTAPI realtime module decls*/
#include "hal.h"         /* HAL public API decls */


#define rtapi_inl inl
#define rtapi_outl outl
#include <asm/io.h>
```

Constantes

Aquí definiremos el offset de cada pin con su nombre según observamos en la tabla, para facilitar el cálculo de su dirección, también tendremos valores de máscaras de un tipo hexadecimal para modificar el bit especifico donde se encuentra los datos del pin, por ejemplo si queremos acceder al 4to bit utilizamos una máscara de “000**1** 0000” que en hexadecimal seria **0x10**.

```
#define GPI4 0x48   //PMIO_48[4]
#define GPI5 0x48   //PMIO_48[5]
            ...
            ...
            ...
#define GPI4MASK 0x10   //PMIO_48[4]
#define GPI5MASK 0x20   //PMIO_48[5]
```

Estructura de Datos

La estructura de datos que definirá a los pines GPIO será llamada viagpio_t.

```
typedef struct{
    WORD pin_addr[8];
    BYTE pin_mask[8];

    hal_bit_t *data_in[8]; // ptrs for input pins 0..7
    hal_bit_t *data_out[4];// ptrs for output pins 0..4
    hal_bit_t data_inv[4]; // polarity params for
                           // output pins 0..4
  long long write_time;
} viagpio_t;
```

Ésta contiene arreglos para las direcciones y máscaras de bits de los 8 pines, un arreglo de punteros que contendrán los datos de los pines de entrada ***data_in**, tanto para su valor normal y una copia del mismo valor pero invertido, un arreglo de punteros con los datos de los pines de salida ***data_out**, y un arreglo con la polaridad (normal (0) o invertida (1)) de los pines de salida.

Función de Lectura de GPI

```
static void read_port(void *arg, long period){
 viagpio_t *port;
 int b;
 int indata[4];

 port = arg;
 //Lee todos los GPI
 indata[0] = rtapi_inl(port->pin_addr[0]);
 indata[1] = rtapi_inl(port->pin_addr[1]);
 indata[2] = rtapi_inl(port->pin_addr[2]);
 indata[3] = rtapi_inl(port->pin_addr[3]);

 /* split the bits into 8 variables (4 regular, 4
inverted) */
 for (b = 0; b < 8; b += 2) {
*(port->data_in[b]) = indata[b/2]& port->pin_mask[b/2];
*(port->data_in[b + 1]) = !(indata[b/2] &
      port->pin_mask[b/2]);
 }
}
```

La función de Lectura recibe como argumento nuestra estructura de datos con la información de los pines, obtenemos los datos leyéndolos de la dirección del pin con la función **inl()** y aislamos el valor del bit específico utilizando su máscara y el operador lógico AND. El resultado y su inverso lo guardamos en el arreglo ***data_in** de la estructura.

Función de Escritura de GPO

En la función de escritura primero leemos y guardamos temporalmente el valor que tiene cada pin inicialmente con la función **inl(),** mediante el uso de la máscara de bits y operadores lógicos modificamos el bit correspondiente a cada pin dependiendo de la salida deseada que se encuentra almacenada en la estructura port:

- Si la salida es 1 y la polaridad 0 (normal) tendremos un 1
- Si la salida es 0 y la polaridad 1 (invertida) tendremos un 1.
- En el resto de casos la salida será 0

Finalmente escribimos estos valores al hardware utilizando la función outl() y la dirección de los pines que tenemos almacenada en nuestra estructura de datos.

```
static void write_port(void *arg, long period)
{
  viagpio_t *port;
  int b;
  BYTE indata[4],  outdata[4];
  port = arg;
  //Lee todos los GPO
  indata[0] = rtapi_inl(port->pin_addr[4]);
                    ...
  for (b = 0; b < 4; b++)
  {
   if ((*(port->data_out[b])) && (!port->data_inv[b]))
      outdata[b] = indata[b] | port->pin_mask[b+4];
   //out:1
      else if ((!*(port->data_out[b])) && (port-
      >data_inv[b]))       outdata[b] = indata[b] |
      port->pin_mask[b+4]; //out:1
   else     //out:0
   outdata[b] = indata[b] & ~(port->pin_mask[b+4]);
  }
  // write it to the hardware
  rtapi_outl(outdata[0], port->pin_addr[4]);
                          ...
  port->write_time = rtapi_get_clocks();
}
```

Funciones Locales

```
static int pins_and_params(void);

static int export_port(viagpio_t * addr);
static int export_input_pin(int pin, hal_bit_t ** base,
int n);
static int export_output_pin(int pin, hal_bit_t **
dbase, hal_bit_t * pbase, int n);
```

La función **pins_and_params()** se encarga de inicializar la mayor parte del controlador, primero crea la componente HAL mediante **hal_init,** luego asigna la memoria compartida para la estructura viagpio_t, también se encarga de guardar los valores

de las direcciones y máscaras de los pines, y a exportar los pines HAL de entrada y salida y asociarlos a los punteros de la estructura mediante la función de exportación **export_port()**, que utiliza la funciones export_input_pin y export_output_pin las cuales se encargan de crear la representación de los pines en la capa HAL con su respectivo identificador y asociándolos con nuestra estructura de datos, por ejemplo: viagpio.pin-01-in, viagpio.pin-05-out, etc.

```
int rtapi_app_main(void)
{
  char name[HAL_NAME_LEN + 1];
  int retval;
                                ...
  // parse "command line", set up pins and parameters
  retval = pins_and_params();
  if (retval != 0)
      return retval;
  /* export functions*/

       /* make read function name */
      rtapi_snprintf(name, sizeof(name),"viagpio.read");
      /* export read function */
      retval = hal_export_funct(name,read_port,
port_data_array, 0, 0, comp_id);
                                ...
      /* make write function name */
     rtapi_snprintf(name, sizeof(name),"viagpio.write");
     /* export write function */
     retval = hal_export_funct(name,write_port,
port_data_array, 0, 0, comp_id);

     rtapi_print_msg(RTAPI_MSG_INFO,
     "MX-PLUSE_GPIO: installed driver for GPIOs\n");
     hal_ready(comp_id);
     return 0;
}

void rtapi_app_exit(void)
{
      hal_exit(comp_id);
}
```

Estas son las funciones main y exit del controlador las cuales se ejecutan en tiempo real, en el main se inicializa el controlador HAL mediante la función pins_and_params() y también se encarga de exportar las funciones de lectura viagpio.read y escritura viagpio.write a la capa HAL.

Compilación

Para agregar nuevos componentes y controladores a LinuxCNC éste debe ser instalado a partir de su código fuente (ver Sección 4.1.1).

Primero ubicaremos nuestro código guardado como **“hal_viagpio.c”** dentro del directorio **"src/hal/drivers/ "** ubicado en la carpeta de LinuxCNC creada por Git.

Abrimos el archivo **“src/Makefile”**, y buscamos la sección que empieza con **“# Subdirectory: hal/drivers”** agregando las siguientes líneas al final:

```
obj-$(CONFIG_HAL_VIAGPIO) += hal_viagpio.o
hal_viagpio-objs    :=    hal/drivers/hal_viagpio.o
$(MATHSTUB)
```

En el mismo archivo buscamos la sección que empieza con **"# Rules to make .o (object) files"** y agregamos al final de todas las líneas que empiezan con ".. /rtlib" lo siguiente:

```
../rtlib/hal_viagpio$(MODULE_EXT):$(addprefix
objects/rt,$(hal_viagpio-objs))
```

Ahora abrimos el archivo **"src/Makefile.inc.in"** y agregamos al final de la sección **"# HAL drivers"** la siguiente línea:

```
CONFIG_HAL_VIAGPIO=m
```

Nos dirigimos al directorio de LinuxCNC y usando el Terminal de Ubuntu ejecutamos los siguientes comandos para compilar:

```
cd src
./Configure
make
```

Con esto ya hemos creado nuestra componente HAL GPIO y ya está lista para ser usada dentro de la configuración de la máquina CNC.

4.3.3 Configuración del archivo de la Capa de Abstracción de Hardware

Como paso final necesitamos inicializar nuestra componente dentro del archivo .hal de la configuración de la máquina,

podemos modificar uno de los archivos generados por el asistente StepConf para facilitar las cosas. Primero, cargamos nuestra componente HAL realtime con la función loadrt y el nombre que definimos en el código “hal_viagpio". Segundo, agregamos nuestras funciones de lectura y escritura a un hilo para que se actualicen a la frecuencia de éste, utilizamos base-thread ya que es el que maneja respuestas rápidas como la generación de pulsos. Por ultimo solo tenemos que enlazar las señales de entrada o salida que queremos utilizar a nuestros pines HAL utilizando el comando net y el identificador del pin.

```
loadrt trivkins
loadrt [EMCMOT]EMCMOT
base_period_nsec=[EMCMOT]BASE_PERIOD
servo_period_nsec=[EMCMOT]SERVO_PERIOD
num_joints=[TRAJ]AXES
loadrt hal_viagpio
loadrt stepgen step_type=0,0,0

addf viagpio.read base-thread
addf stepgen.make-pulses base-thread
addf viagpio.write base-thread
                ...
net xdir => viagpio.pin-05-out
net xstep => viagpio.pin-06-out
net zdir => viagpio.pin-07-out
net zstep => viagpio.pin-08-out
               ...
```

Podemos encontrar este archivo en el anexo E.

CAPÍTULO 5

5. PRUEBAS Y RESULTADOS

5.1 Prueba de Señales de Salida Puerto Paralelo

Se realizó la prueba de las señales de salida del puerto paralelo con ayuda de un osciloscopio y utilizando LinuxCNC y los botones que permiten mover cada eje:

- **Eje X:** Direccionales Izquierda/Derecha
- **Eje Y:** Direccionales Arriba/Abajo
- **Eje Z:** Teclas Re Pág. /Av. Pág.

También se utilizó el control de velocidad en la interfaz de LinuxCNC para observar los cambios en la frecuencia de los pulsos al variar este parámetro.

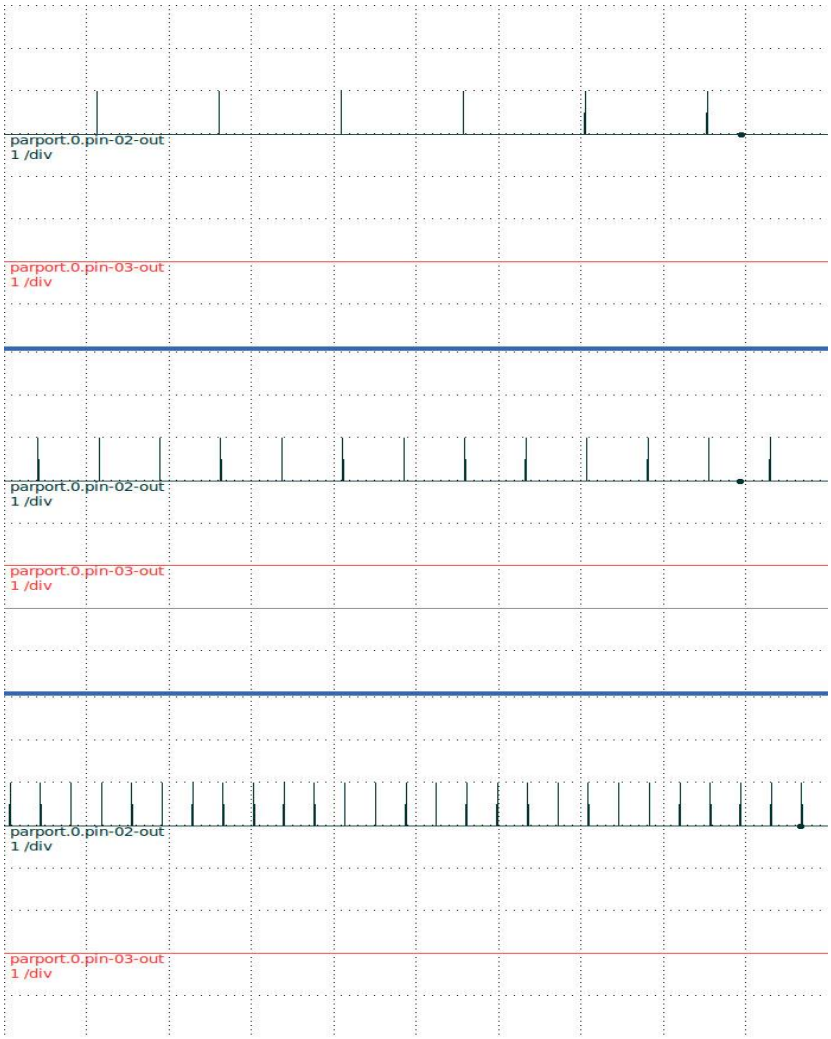


Figura 5.1 *Señales de Salida a 50, 100, y 200 mm/s*

En la **Figura 5.1** se muestran las señales de salida de los pines 02 y 03 correspondientes a la señal de dirección y la de pasos del eje X. La primera imagen corresponde a una velocidad de 50mm/s moviéndose a la derecha, la segunda a una velocidad de 100mm/s moviéndose a la izquierda y la última a una velocidad de 200mm/s moviéndose a la derecha.

5.2 Pruebas de los Motores de paso

Pruebas de Velocidad y Aceleración Máxima

Durante la creación del archivo de configuración con Stepconf se necesitaba especificar las velocidades y aceleración máximas de cada motor, ahora utilizaremos la herramienta de prueba (ver **Figura 5.2**) que se encontraba en esa sección usando el botón “Test this axis”.

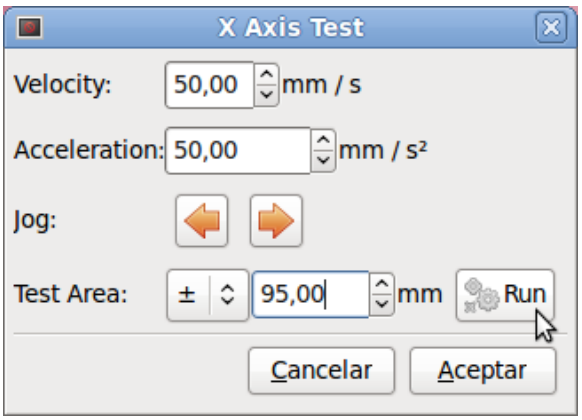


Figura 5.2 *Herramienta de Prueba de Ejes*

El procedimiento consiste en ubicar la herramienta (en el caso de la impresora el extrusor) en una posición central al eje de prueba y marcar la posición. Luego definiremos un área de prueba en la que se moverá la herramienta, el área de trabajo dela impresora es 200x200x100, pero usaremos un valor menor por seguridad, para X (±95), Y (±95), Z (±45).

Iniciamos con valores bajos de velocidad y aceleración, al ejecutar el test, la herramienta comenzara a moverse a ambos extremos del eje a la velocidad utilizada, la detenemos y la herramienta debería volver a su posición inicial, si la posición es incorrecta significa que el motor se atascó o perdió pasos y reducimos la velocidad, en caso contrario podemos probar aumentándola. Una vez encontrada la velocidad máxima basados en el margen de error obtenido, la reducimos un 10% por seguridad y la asignamos a la máquina.

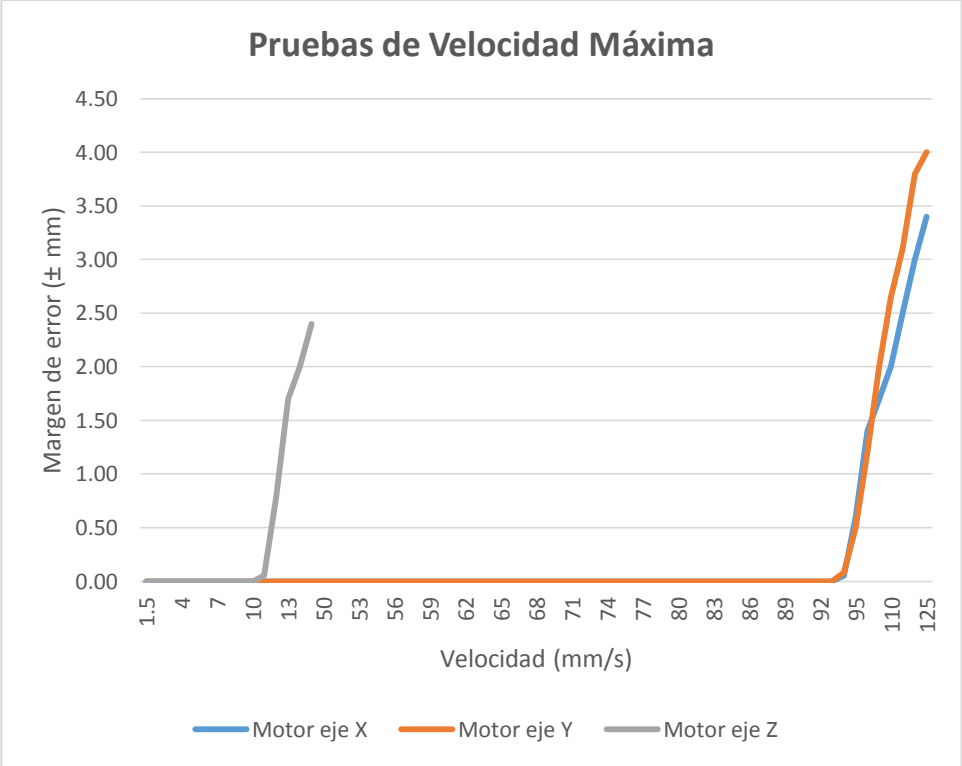


Figura 5.3 *Experimento - Pruebas de velocidad de motores*

En la Figura 5.3, se muestran los resultados del experimento realizado para medir la velocidad máxima, se puede observar comparando los resultados, que en los ejes X, Y, a medida que la velocidad supera alrededor de los 100mm/s los motores empezaban a atascarse y no podían regresar a su posición inicial. El eje Z es algo especial ya que consiste de 2 motores y utilizan un husillo para su movimiento, las velocidades utilizadas son menores considerando que se desplaza aproximadamente 1.4 mm/revolución, se puede observar que empezaba a fallar a medida que la velocidad se acercaba a 15mm/s.
Considerando estos resultados, las velocidades máximas para cada eje serían las siguientes:

Eje X = 90mm/s **Eje Y** = 90mm/s **Eje Z** = 12mm/s

Ajustándolas al valor de velocidad recomendado obtendríamos:

Eje X = 81mm/s **Eje Y** = 81mm/s **Eje Z** = 10mm/s

Una vez obtenida la velocidad máxima, es turno de la aceleración mediante el mismo procedimiento, pero esta vez ya conociendo la velocidad máxima esta vez empezamos a variar la aceleración.

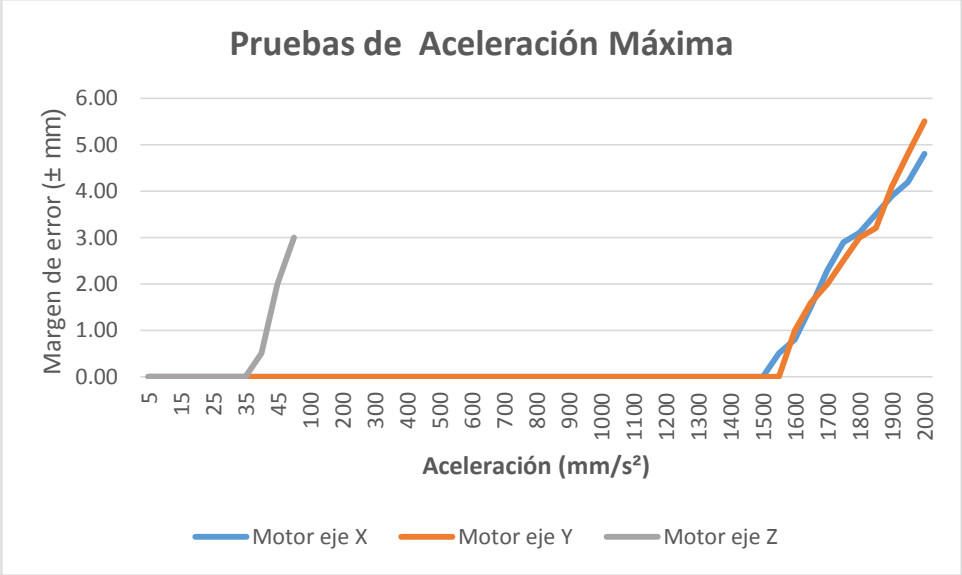


Figura 5.4 *Experimento - Pruebas de aceleración de motores*

En la Figura 5.4 se muestran los resultados del experimento para encontrar la aceleración máxima, en el eje X, Y se observa que ambos ejes comenzaban a tener errores a medida que se acercaban a los 1600 mm/s^2, en el eje Z se manejaban aceleraciones menores, se observó que a medida que se acercaba a los 45 mm/s^2 empezaban a tener fallos.

En base a las pruebas se determinaron las aceleraciones máximas:

Eje X = 1500mm/s^2 **Eje Y** = 1500mm/s^2 **Eje Z** = 40mm/s^2

Ajustando las aceleraciones a su valor recomendado resultaría en:

Eje X = 1350mm/s^2 **Eje Y** = 1350mm/s^2 **Eje Z** = 35mm/s^2

5.3 Prueba del Sistema ejecutando Código G

Con la velocidad y aceleración de cada eje correctamente definida procedemos a realizar las pruebas del sistema CNC junto a la impresora ejecutando el código-G que generamos en la sección 4.2.4 en LinuxCNC.

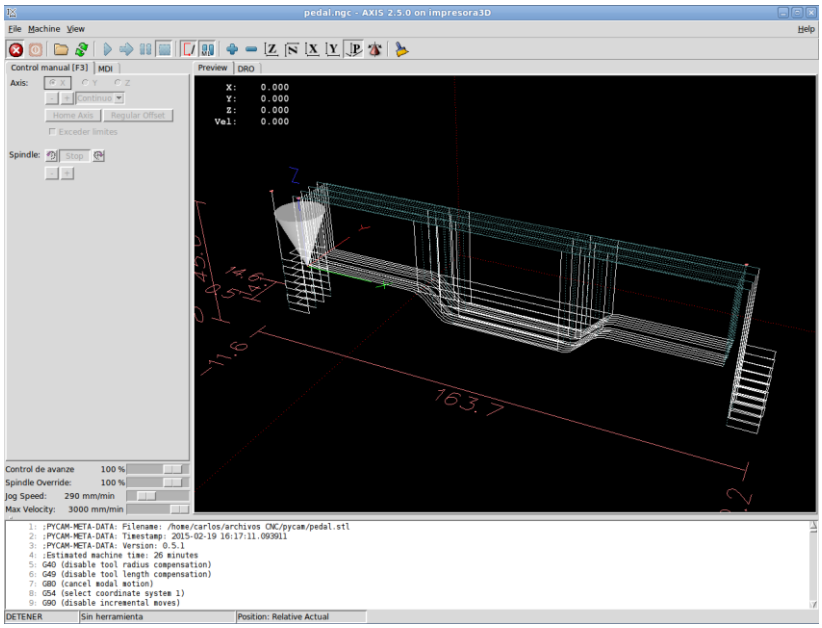


Figura 5.5 *LinuxCNC con el codigo-G generado*

Figura 5.6 *Impresora en funcionamiento*

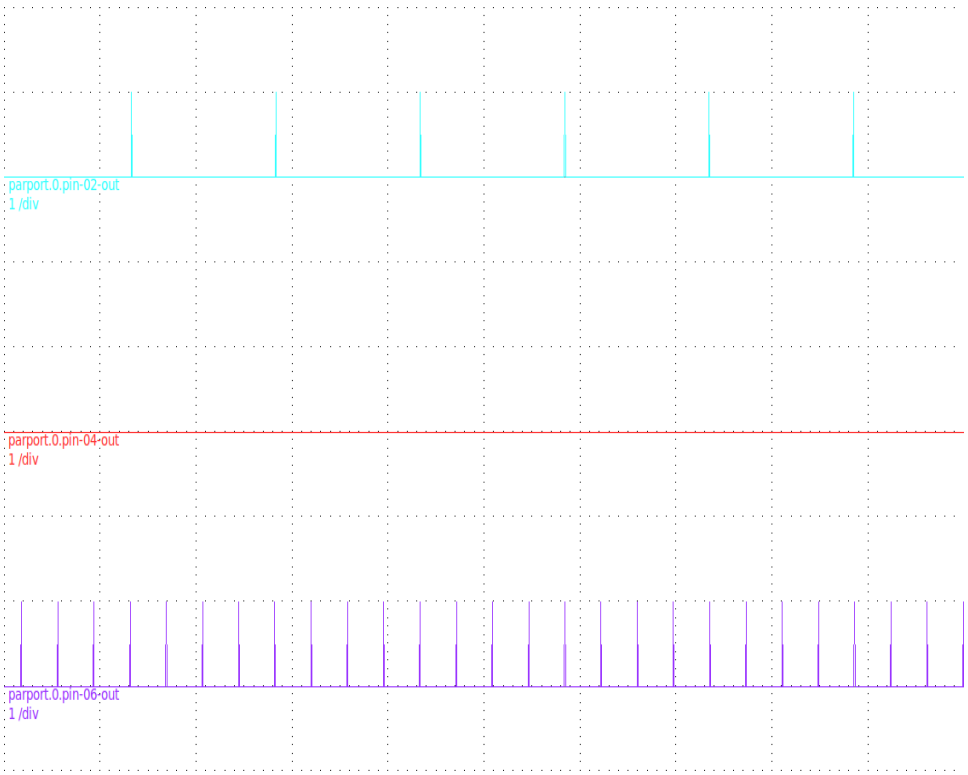


Figura 5.7 *Señales de Paso de los ejes XYZ*

La Figura 5.5 muestra la pre-visualización del código-G en LinuxCNC, la ejecución se realizó de forma correcta, no hubo ningún problema en el movimiento de los motores de la impresora como se ve en la Figura 5.6, y la Figura 5.7 muestra que tampoco hubo problema en las señales de salida del puerto.

5.4 Prueba del Controlador GPIO diseñado

Prueba de Señales de Salida en los pines GPIO

Una vez compilado el controlador que creamos y definida la configuración de la máquina (ver sección 4.3), realizamos la prueba de las señales de salida de los pines GPO utilizando un osciloscopio.

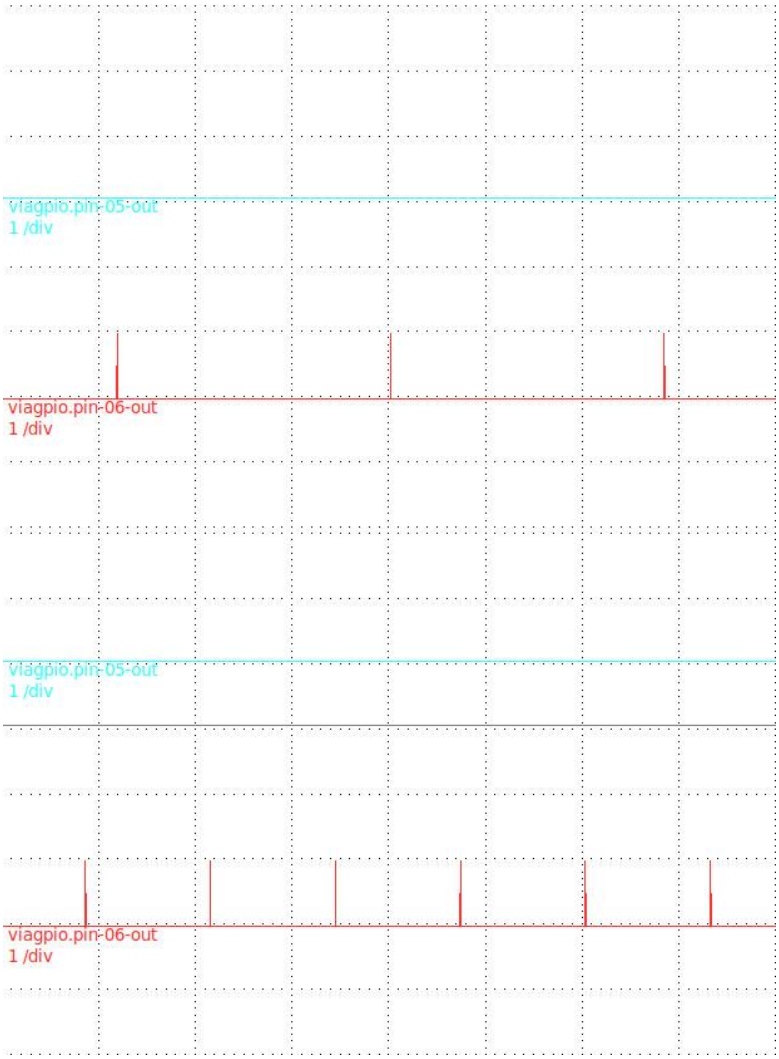


Figura 5.8 *Senales de Salida de los pines GPIO*

La Figura 5.8 muestra las señales de dirección en el pin viagpio.pin-05-out (GPO12) y la señal de paso del eje X en el pin viagpio.pin-06-out (GPIO01). En la primera imagen el movimiento es hacia la derecha (dir=Low) y en la segunda hacia la izquierda (dir=High), ambas con diferentes velocidades.

5.5 Análisis de Resultados

En la prueba 5.1 se observaron las señales de salida de dirección y de paso de los motores vía puerto paralelo, se experimentó con varias velocidades y cambios de dirección para observar el comportamiento de las señales, obteniendo excelentes resultados como se muestra en la Figura 5.1 con velocidades de 50mm/s y 100mm/s.

En la prueba 5.2 sobre las velocidades y aceleraciones máximas de los motores, se determinó observando la Figura 5.3 y la Figura 5.4 que para los ejes X, Y, que funcionan de forma similar, su velocidad máxima estaba en los 90mm/s y su aceleración máxima en 1500mm/s^2, con los valores recomendados de 81mm/s de velocidad y 1350mm/s^2 de aceleración, en el eje Z conformado por dos motores paralelos que utilizan un husillo para su movimiento, las velocidad

máxima era de 12 mm/s y aceleración de 40mm/s^2, y los valores recomendados de 10mm/s y 35mm/s^2 respectivamente.

Los resultados de la prueba 5.3 mostraron el correcto funcionamiento de la impresora 3D controlada mediante LinuxCNC ejecutando código G para su movimiento.

Los resultados de la prueba 5.4 para observar las señales de salida por medio de los pines GPIO demostró que no hay ninguna diferencia comparado a una salida vía puerto paralelo como se muestra en la Figura 5.8, siendo posible obtener un mayor número de posibilidades de conexión por medio de este medio.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1) Las pruebas realizadas y la baja latencia obtenida, demostraron que un Sistema embebido comercial como el que se utilizó, cumple perfectamente el rol de un Sistema de Control CNC utilizando el software adecuado, además de que tiene especificaciones técnicas que se pueden encontrar normalmente.

2) Utilizar un Sistema Embebido o una PC también nos da la ventaja de poder utilizar las otras herramientas necesarias en la fase de diseño como lo son el software CAD/CAM logrando así tener un Sistema CNC totalmente completo, siempre y cuando se cuente con los requerimientos necesarios.

3) El Sistema Linux junto al software LinuxCNC demostraron ser un excelente medio como Sistema de control CNC, al utilizar un kernel de

Linux modificado con la extensión RTAI garantiza que la generación de pulsos sea precisa y a tiempo, por lo que no es necesario de hardware adicional para este propósito como en otros sistemas.

4) El archivo de configuración de LinuxCNC desarrollado para la impresora 3D permitió que esta trabajara como cualquier maquinaria CNC de 3 ejes, su funcionamiento fue excelente durante las pruebas, esto nos demuestra que con los parámetros adecuados en la configuración se puede operar cualquier máquina-herramienta CNC, e incluso cualquier máquina que utilice motores de paso sin problemas.

5) El controlador GPIO desarrollado funciono adecuadamente, se obtuvo el mismo resultado comparándolo con el puerto paralelo, esto demuestra que se puede adaptar el Sistema para que aproveche estos pines GPIO como recurso para controlar otros dispositivos, sensores, botones, sin requerir otro puerto paralelo.

Recomendaciones

Este tipo de proyectos tiene un gran potencial y muchas aplicaciones, aquí se detallaran algunas recomendaciones para futuras investigaciones de la misma temática.

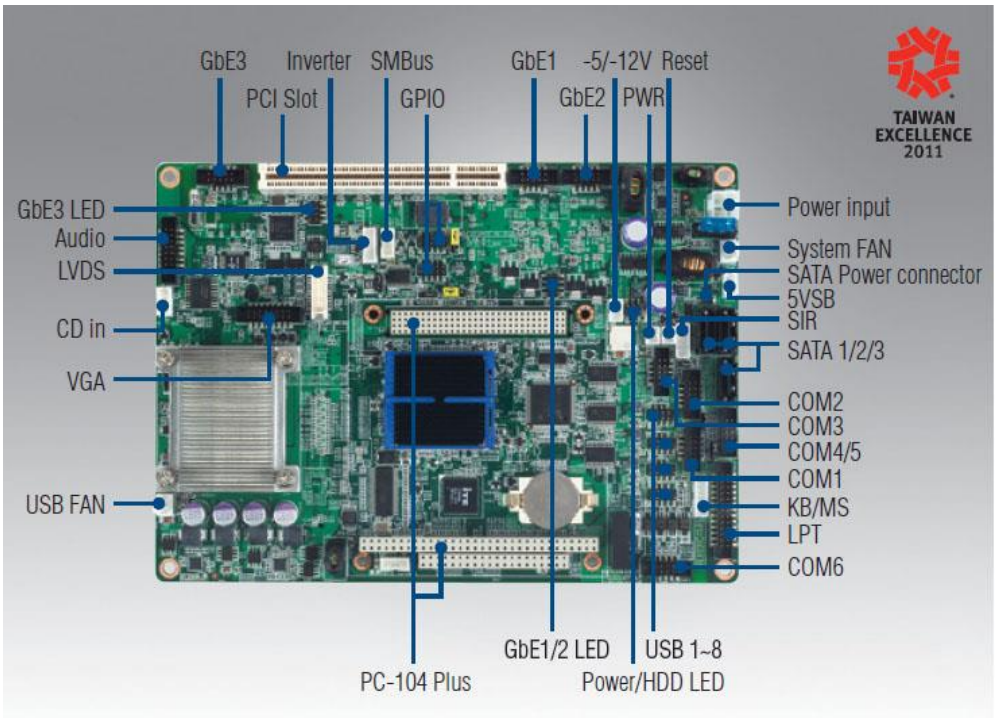
1) En este proyecto no se cubrió el uso de servo motores por lo que podría ser de utilidad explorar las características y varias ventajas que estos poseen frente a los motores de paso. Otro caso de interés es el uso de máquina-herramientas de rotación como el torno.

2) Investigar y aprovechar los beneficios que ofrece el uso de software libre, LinuxCNC ofrece una gran cantidad de alternativas de configuración, es posible crear una interfaz de usuario personalizada de acuerdo a nuestras necesidades, incluso utilizar joysticks para el movimiento manual de los ejes, además de que LinuxCNC cuenta con una gran comunidad de desarrolladores que aporta continuamente con nuevas funcionalidades.

3) Considerar la posibilidad de optimizar o mejorar el funcionamiento del LinuxCNC, se podría crear un método de control utilizando laser o cámaras para mejorar la precisión de la herramienta. Esta es otra de las libertades que ofrece usar software libre.

ANEXOS

Anexo A: Hoja de Datos del Embebido Advantech PCM-9562

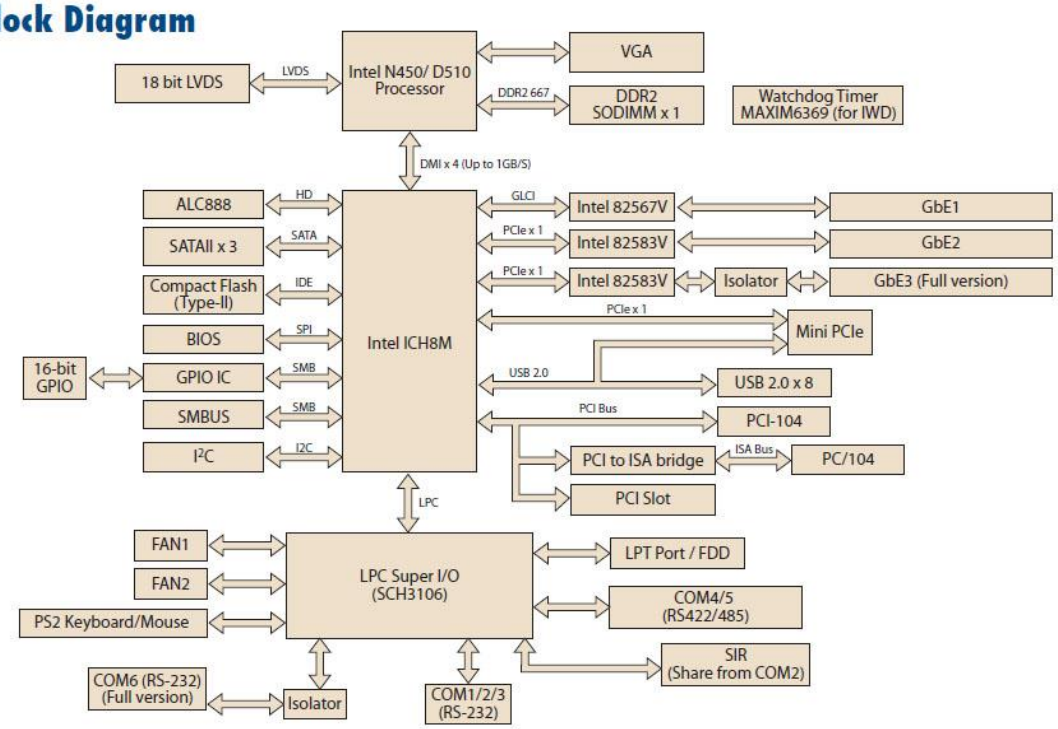


Processor System	CPU	Intel Atom N450/D510 1.66 GHz
	Frequency	Atom N450/D510 1.66 GHz
	L2 Cache	512 KB/1 MB
	System Chipset	N450/D510 + ICH8M
	BIOS	AMI 16 Mbit
Memory	Technology	DDR2 667 MHz
	Max. Capacity	2 GB
	Socket	1 x 200-pin SODIMM
Display	Chipset	N450/D510
	VRAM	Optimized Shared Memory Architecture up to 224 MB
	Graphics Engine	Embedded Gen3.5+ GFX Core, HW MPEG2 decoder
	LVDS	Single channel 18-bit LVDS up to WXGA 1366 x 768
	VGA	N450: Up to SXGA 1400 x 1050 @ 60 Hz (SXGA) D510: up to 2048 x 1536 (QXGA)
	Dual Display	VGA+ LVDS
Ethernet	Interface	3 (RJ-45 connector through the cable and Ethernet3 is Full version only)
	Controller	GbE1 Intel 82567, GbE2 Intel 82583V, GbE3 Intel 82583V (UL60601 Compliant)
	Connector	Box header

Audio	Chipset	ALC888 HD Codec, Speaker out, CD-input, Line-in, Line-out, Mic-in
	Amplifier	Max 2.2W/ch Stereo into a 3Ω Load
WatchDog Timer	Output	System reset
	Internal	Watchdog timer1 (IWT): monitor the system status before OS is ready (programmable 10ms, disable, 1s, 60s) Watchdog timer2 (PWT): monitor the application status after OS is ready (programmable 1 - 255 sec/min) (Full version only)
Storage	CompactFlash	CompactFlash type I/II
	SATA	3 SATA II (Max. Data Transfer Rate 300 MB/s)
	Floppy	Share with LPT (Optional)
	SPI Flash	16 Mbit
Internal I/O	Serial	4 x RS-232 (COM1/2/3/6, isolation design in COM6 is Full version only) 2 x RS-422/485 (COM4/5, default RS-422/485, RS-232 with TX/RX only is optional by request) ESD protection for RS-232: Air gap ±15kV, Contact ±8kV
	Ethernet	GbE x 3 (RJ-45 connector through the cable and GbE3 is Full version)
	PS/2 KB/Mouse	1
	VGA	1
	Reset Button	1
	USB	8 x USB 2.0
	Parallel (LPT)	1
	FDD	Share with LPT (Optional)
	GPIO	16-bit GPIO
	SMBUS	1
	I2C	1
Expansion	PC/104-Plus slot	1
	Full-size Mini PCIe	1
	PCI Slot	1
Power	Power Type	AT / ATX (Both AT/ATX can support ACPI)
	Power Supply Voltage	ATX: 12V ±10%, 5VSB ±5% (5V stand-by power is only for auto power off function) AT: 12V ±10% only
	Power Consumption (Typical)	CM-9562N-S6A1E: 893 mA @ 12 V, 8 mA @ 5 VSB (10.8 W) PCM-9562D-S6A1E: 1130 mA @ 12 V , 10 mA @ 5 VSB (13.6 W)
	Power	PCM-9562N-S6A1E: 1159 mA @ 12 V, 6 mA @ 5 VSB

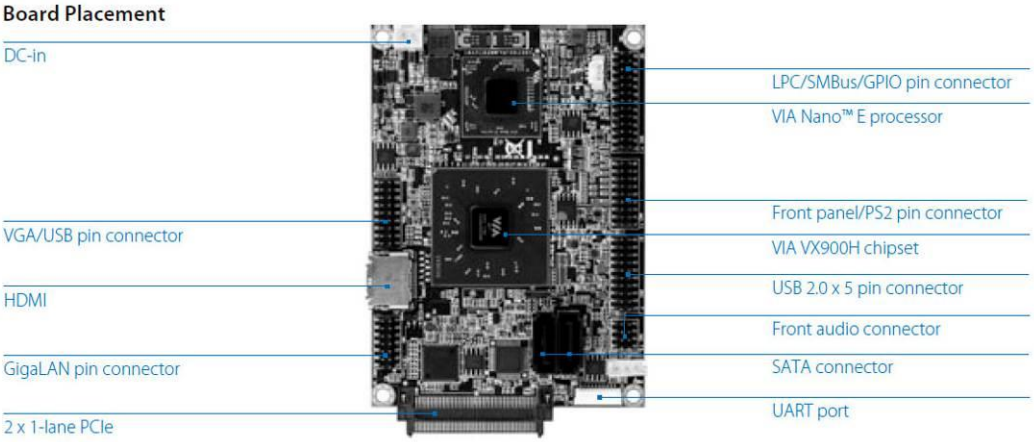
	Consumption	(13.9 W) (Max, test in HCT) PCM-9562D-S6A1E: 1404 mA @ 12 V, 8 mA @ 5 VSB (16.9 W)
Environment	Operating	0 ~ 60° C (32 ~ 140° F) (Operating humidity: 40° C @ 95% RH non-condensing)
	Non-Operating	-40° C ~ 85° C and 60° C @ 95% RH non-condensing
Physical Characteristics	Dimensions (L x W)	203 x 146 mm (8" x 5.75")
	Weight	0.7 kg (1.54 lb) (with heatsink)
	Total Height (with cooler + PCB + Bottom)	29.6mm (PCM-9562N/NF), 38.6mm (PCM-9562D/DF), 44.6mm (PCM-9562Z/Z2)

Block Diagram



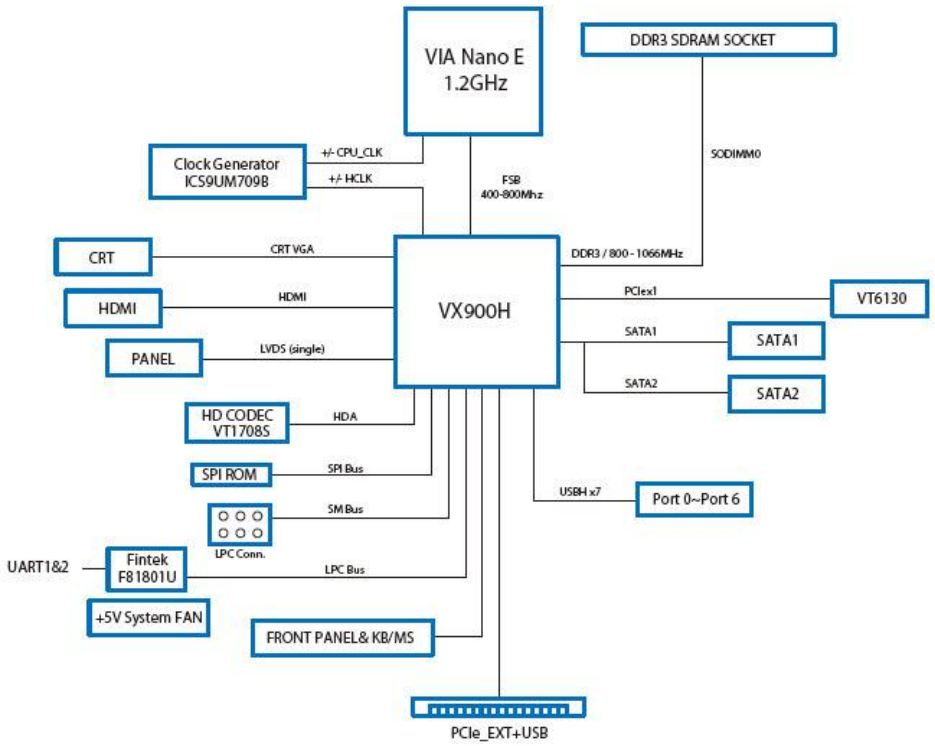
Anexo B: Hoja de datos del Embebido VIA EPIA-P380 [16]

Board Placement



Model Name	EPIA-P830-12L
Processor	1.2GHz VIA Nano™ E
Chipset	VIA VX900H Media System Processor
System Memory	1 x DDR3 800/1066 SODIMM socket Up to 4GB memory size
VGA	Integrated VIA Chrome™9 HD DX9 3D/2D graphics with VC1, MPEG-2, WMV9 and H.264 decoding acceleration
Onboard Serial ATA	2 x SATA connectors
Onboard LAN	1 x VIA VT6130 PCIe Gigabit Ethernet controller
Onboard Audio	VIA VT1708S High Definition Audio Codec
Onboard Super IO	Fintek F81801U-I
Onboard I/O Connectors	1 x USB pin connector for 5 additional USB 2.0 ports 1 x LPC pin connector 1 x SMBus pin connector 1 x Front panel pin connector 1 x PS2 mouse/keyboard pin connector 1 x Fan pin connectors: Sys FAN 1 x 1-CH 24-bit LVDS panel pin connector 1 x Audio pin connector for Line-in, Line-out & MIC-in 1 x LAN pin connector 1 x DIO pin connector (4GPI+4GPO) 2 x UART port pin-headers 1 x Backlight control for panel 1 x SPI flash connector/VCP 1 x SATA power connector

	1 x 12V DC-in power connector
Back Panel I/O	1 x HDMI port 1 x VGA port 1 x GigaLAN port 2 x USB 2.0 ports
BIOS	AMI BIOS 8Mbit SPI flash memory
Operating System	Windows 7, Windows Embedded CE, Windows Embedded Standard, Windows XP, Linux
System Monitoring & Management	Wake-on LAN, Keyboard Power-on, Timer Power-on System power management, AC power failure recovery Watch Dog Timer
Operating Temperature	0°C up to 60°C
Operating Humidity	0% ~ 95% (relative humidity; non-condensing)
Form Factor	Pico-ITX 10 cm x 7.2 cm (3.9" x 2.8")
Compliance	CE/FCC/BSMI/RoHS



Anexo C: Archivo de Configuración .INI de la Impresora 3D

```
[EMC]
MACHINE = impresora3D
DEBUG = 0

[DISPLAY]
DISPLAY = axis
EDITOR = gedit
POSITION_OFFSET = RELATIVE
POSITION_FEEDBACK = ACTUAL
MAX_FEED_OVERRIDE = 1.2
INTRO_GRAPHIC = linuxcnc.gif
INTRO_TIME = 5
PROGRAM_PREFIX = /home/carlos/linuxcnc/nc_files
INCREMENTS = 5mm 1mm .5mm .1mm .05mm .01mm .005mm

[FILTER]
PROGRAM_EXTENSION = .png,.gif,.jpg Greyscale Depth Image
PROGRAM_EXTENSION = .py Python Script
png = image-to-gcode
gif = image-to-gcode
jpg = image-to-gcode
py = python

[TASK]
TASK = milltask
CYCLE_TIME = 0.010

[RS274NGC]
PARAMETER_FILE = linuxcnc.var

[EMCMOT]
EMCMOT = motmod
COMM_TIMEOUT = 1.0
COMM_WAIT = 0.010
BASE_PERIOD = 100000
SERVO_PERIOD = 1000000

[HAL]
HALFILE = impresora3D.hal
HALFILE = custom.hal
POSTGUI_HALFILE = custom_postgui.hal

[TRAJ]
AXES = 3
COORDINATES = X Y Z
LINEAR_UNITS = mm
ANGULAR_UNITS = degree
CYCLE_TIME = 0.010
DEFAULT_VELOCITY = 9.00
MAX_LINEAR_VELOCITY = 90.00
```

```
[EMCIO]
EMCIO = io
CYCLE_TIME = 0.100
TOOL_TABLE = tool.tbl

[AXIS_0]
TYPE = LINEAR
HOME = 0.0
MAX_VELOCITY = 90.0
MAX_ACCELERATION = 1350.0
STEPGEN_MAXACCEL = 1500.0
SCALE = 4.0
FERROR = 1
MIN_FERROR = .25
MIN_LIMIT = -50.0
MAX_LIMIT = 200.0
HOME_OFFSET = 0.0

[AXIS_1]
TYPE = LINEAR
HOME = 0.0
MAX_VELOCITY = 90.0
MAX_ACCELERATION = 1350.0
STEPGEN_MAXACCEL = 1500.0
SCALE = 4.0
FERROR = 1
MIN_FERROR = .25
MIN_LIMIT = -50.0
MAX_LIMIT = 200.0
HOME_OFFSET = 0.0

[AXIS_2]
TYPE = LINEAR
HOME = 0.0
MAX_VELOCITY = 10.0
MAX_ACCELERATION = 35.0
STEPGEN_MAXACCEL = 40.0
SCALE = 285.714285714
FERROR = 1
MIN_FERROR = .25
MIN_LIMIT = -100.0
MAX_LIMIT = 100.0
HOME_OFFSET = 0.0
```

Anexo D: Código del Controlador GPIO

```
/********************************************************************
* Description:  hal_viagpio.c
*               This file, 'hal_viagpio.c', is a HAL component that
*               provides a driver for the VIA EPIA-P830 GPIO.
*
* Author: Carlos Ronquillo Castro
*
********************************************************************/

#include "rtapi.h"          /* RTAPI realtime OS API */
#include "rtapi_ctype.h"        /* isspace() */
#include "rtapi_string.h"       /*strcmp() */
#include "rtapi_math.h"
#include "rtapi_app.h"          /* RTAPI realtime module decls */
#include "hal.h"            /* HAL public API decls */

#define rtapi_inl inl
#define rtapi_outl outl
#include <asm/io.h>

//CONSTANTES
#define GPI4 0x48   //PMIO_48[4]
#define GPI5 0x48   //PMIO_48[5]
#define GPI8 0x49   //PMIO_49[0]
#define GPI9 0x49   //PMIO_49[1]

#define GPO12 0x4F  //PMIO_4F[2]
#define GPIO1 0x4D  //PMIO_4D[4]
#define GPIO9 0x4E  //PMIO_4E[4]
#define GPIO12 0x4E //PMIO_4E[7]

#define GPI4MASK 0x10   //PMIO_48[4]
#define GPI5MASK 0x20   //PMIO_48[5]
#define GPI8MASK 0x01   //PMIO_49[0]
#define GPI9MASK 0x02   //PMIO_49[1]

#define GPO12MASK 0x04  //PMIO_4F[2]
#define GPIO1MASK 0x10  //PMIO_4D[4]
#define GPIO9MASK 0x10  //PMIO_4E[4]
#define GPIO12MASK 0x80 //PMIO_4E[7]


#define VX900 0x80008800
typedef uint16_t WORD;
typedef uint32_t DWORD;
typedef uint8_t  BYTE;


WORD wPmioBase;
```

```
/* module information */
MODULE_AUTHOR("CVR-ESPOL");
MODULE_DESCRIPTION("VIA EPIA P830 GPIO Driver for LinuxCNC HAL");
MODULE_LICENSE("GPL");

/***********************************************************************
*                STRUCTURES AND GLOBAL VARIABLES
************************************************************************/

/* this structure contains the runtime data needed by the
   GPIO driver for a single port
*/
typedef struct
{
    WORD pin_addr[8];       /* direcciones de los pines fisicos */
    BYTE pin_mask[8];       /* mascaras para modificar bit del pin*/

/* punteros a pines de entrada 4 regulares/4 invertidos*/
    hal_bit_t *data_in[8];
    hal_bit_t *data_out[4];    /* punteros a pines de salida */
    hal_bit_t data_inv[4];     /* polaridad pines de salida */

    long long write_time;
} viagpio_t;

/* pointer to array of viagpio_t structs in shared memory */
static viagpio_t *port_data_array;

/* other globals */
static int comp_id;     /* component ID */

static unsigned long ns2tsc_factor;
#define ns2tsc(x) (((x) * (unsigned long long)ns2tsc_factor) >> 12)

/***********************************************************************
*                  LOCAL FUNCTION DECLARATIONS
************************************************************************/

/* These are the functions that actually do the I/O, everything else
is just init code */

static void read_port(void *arg, long period);
static void write_port(void *arg, long period);


/* 'pins_and_params()' does most of the work involved in setting up
the driver.   It parses the command line (argv[]), then if the
command line is OK, it calls hal_init(), allocates shared memory for
the viagpio_t data structure(s), and exports pins and parameters. It
does not set up functions, since that is handled differently in
realtime and user space. */
static int pins_and_params(void);

static int export_port(viagpio_t * addr);
```

```
static int export_input_pin(int pin, hal_bit_t ** base, int n);
static int export_output_pin(int pin, hal_bit_t ** dbase, hal_bit_t
* pbase, int n);


/***********************************************************************
 *   GetPMIOBaseAddr
************************************************************************/
void GetPMIOBaseAddr(void);

/***********************************************************************
* Enable and initialize GPO port (eg. Enable GPO)
************************************************************************/
void GPOEnable(void);


/***********************************************************************
*                                                  INIT  AND  EXIT  CODE
*
************************************************************************/

int rtapi_app_main(void)
{

    char name[HAL_NAME_LEN + 1];
    int retval;

#if LINUX_VERSION_CODE > KERNEL_VERSION(2,6,0)
    // this calculation fits in a 32-bit unsigned
    // as long as CPUs are under about 6GHz
    ns2tsc_factor = (cpu_khz << 6) / 15625ul;
#else
    ns2tsc_factor = 1ll<<12;
#endif

    /* parse "command line", set up pins and parameters */
    retval = pins_and_params();
    if (retval != 0)
        return retval;
    /* export functions*/

        /* make read function name */
        rtapi_snprintf(name, sizeof(name), "viagpio.read");
        /* export read function */
        retval = hal_export_funct(name, read_port, port_data_array,
                                  0, 0, comp_id);
        if (retval != 0)
        {
            rtapi_print_msg(RTAPI_MSG_ERR, "MX-PLUSE_GPIO: ERROR:
                            Port read funct export failed\n");
            hal_exit(comp_id);
            return -1;
        }
```

```
        /* make write function name */
        rtapi_snprintf(name, sizeof(name), "viagpio.write");
        /* export write function */
        retval = hal_export_funct(name, write_port, port_data_array,
                                  0, 0, comp_id);
        if (retval != 0)
        {
            rtapi_print_msg(RTAPI_MSG_ERR,"MX-PLUSE_GPIO: ERROR:
                            port write funct export failed\n");
            hal_exit(comp_id);
            return -1;
        }

    rtapi_print_msg(RTAPI_MSG_INFO,
    "MX-PLUSE_GPIO: installed driver for GPIOs\n");
    hal_ready(comp_id);
    return 0;
}


void rtapi_app_exit(void)
{
    hal_exit(comp_id);
}


/***********************************************************************
* inl(): IN data from I/O port by long word 32-bit
* outl(): OUT data to I/O port by long word 32-bit
***********************************************************************/

/***********************************************************************
* GetPMIOBaseAddr */
***********************************************************************/
void GetPMIOBaseAddr(void)
{
    DWORD pciAddr;
    /*
    * For (VX900)
    * PMIO Base Address is located in PCI configuration space
    * Bus 0, device 17, function 0, offset 0x88
    */
    pciAddr = VX900 | 0x88;
    outl(pciAddr,0x0CF8);
    wPmioBase = inl(0x0CFC);
    wPmioBase &= 0xFFFE;
    //printf("PMIO Base address = %x\n", wPmioBase);
} /* end GetPMIOBaseAddr */
```

```
/*****************************************************************
* Enable and initialize GPO ports
*****************************************************************/
void GPOEnable(void)
{
    DWORD pciAddr;
    DWORD value;
//GPO12
    /*
    * For (VX900)
    * GPO Control is located in PCI configuration space
    * Bus 0, device 17, function 0, offset 0x9B
    */
    pciAddr = VX900 | 0x9B;
    outl(pciAddr, 0x0CF8);
    value = inl(0x0CFC);
    /*
    * eg. Enable GPO 12
    * GPO 12: Rx9B[0] = 1
    */
    value |= 0x01000000;  /* set 1 bit 24??? */
    outl(value, 0x0CFC);


//GPIO1
    pciAddr = VX900 | 0x95;
    outl(pciAddr, 0x0CF8);
    value = inl(0x0CFC);

     //eg. Enable GPIO1
     //PMU_RX95[3]=1
     //PMU_RX95[2]=1

    value |= 0x0000000c; // set 1 to bit 2 and 3 : Hex C = 1100
    outl(value, 0x0CFC);


//GPIO12
    pciAddr = VX900 | 0x80;
    outl(pciAddr, 0x0CF8);
    value = inl(0x0CFC);

     //eg. Enable GPIO12
     //PMU_RX80[6]=0

    value &= 0xFFFFFFBF;  //set 0 to bit 6
    outl(value, 0x0CFC);

} /* end GPOEnable */
```

```
/***********************************************************************
*                  REALTIME PORT READ AND WRITE FUNCTIONS
************************************************************************/
static void read_port(void *arg, long period)
{
    viagpio_t *port;
    int b;
    int indata[4];

    port = arg;
    //Lee todos los GPI
    indata[0] = rtapi_inl(port->pin_addr[0]);
    indata[1] = rtapi_inl(port->pin_addr[1]);
    indata[2] = rtapi_inl(port->pin_addr[2]);
    indata[3] = rtapi_inl(port->pin_addr[3]);

    /* Divide los bits en 8 variables (4 regulares, 4 invertidas)*/
    for (b = 0; b < 8; b += 2)
    {
        *(port->data_in[b]) = indata[b/2] & port->pin_mask[b/2];
        *(port->data_in[b + 1]) = !(indata[b/2] &
            port->pin_mask[b/2]);
    }
}

static void write_port(void *arg, long period)
{
    viagpio_t *port;
    int b;
    BYTE indata[4],  outdata[4];

    port = arg;

    //Lee todos los GPO
    indata[0] = rtapi_inl(port->pin_addr[4]);
    indata[1] = rtapi_inl(port->pin_addr[5]);
    indata[2] = rtapi_inl(port->pin_addr[6]);
    indata[3] = rtapi_inl(port->pin_addr[7]);

    // Asigna el valor de salida
    for (b = 0; b < 4; b++)
    {
        // get the data, add to output byte
        //Si la salida es 1 y polaridad 0 -> salida = 1
        if ((*(port->data_out[b])) && (!port->data_inv[b]))
        outdata[b] = indata[b] | port->pin_mask[b+4];
        //Si la salida es 0 y polaridad 1(invertido) -> salida = 1
        else if ((!*(port->data_out[b])) && (port->data_inv[b]))
        outdata[b] = indata[b] | port->pin_mask[b+4];
        //Salida = 0
        else
        outdata[b] = indata[b] & ~(port->pin_mask[b+4]);
    }
```

```
    // Escribir en los pines fisicos
    rtapi_outl(outdata[0], port->pin_addr[4]);
    rtapi_outl(outdata[1], port->pin_addr[5]);
    rtapi_outl(outdata[2], port->pin_addr[6]);
    rtapi_outl(outdata[3], port->pin_addr[7]);

    port->write_time = rtapi_get_clocks();
}

/***********************************************************************
*                   LOCAL FUNCTION DEFINITIONS
************************************************************************/

static int pins_and_params(void)
{
    WORD pin_addr[8];
    BYTE pin_mask[8];

    int i, retval;

    GetPMIOBaseAddr();
    GPOEnable();

    //Inputs
    pin_addr[0] = wPmioBase + GPI4;
    pin_addr[1] = wPmioBase + GPI5;
    pin_addr[2] = wPmioBase + GPI8;
    pin_addr[3] = wPmioBase + GPI9;
    //Outputs
    pin_addr[4] = wPmioBase + GPO12;
    pin_addr[5] = wPmioBase + GPIO1;
    pin_addr[6] = wPmioBase + GPIO9;
    pin_addr[7] = wPmioBase + GPIO12;

    //Inputs
    pin_mask[0] = GPI4MASK;
    pin_mask[1] = GPI5MASK;
    pin_mask[2] = GPI8MASK;
    pin_mask[3] = GPI9MASK;
    //Outputs
    pin_mask[4] = GPO12MASK;
    pin_mask[5] = GPIO1MASK;
    pin_mask[6] = GPIO9MASK;
    pin_mask[7] = GPIO12MASK;


    /* have good config info, connect to the HAL */
    comp_id = hal_init("hal_viagpio");
    if (comp_id < 0)
    {
    rtapi_print_msg(RTAPI_MSG_ERR, "MX-PLUSE_GPIO: ERROR: hal_init()
                    failed\n");
    return -1;
    }
```

```
    /* allocate shared memory for viagpio data */
    port_data_array = hal_malloc(sizeof(viagpio_t));

    if (port_data_array == 0)
    {
    rtapi_print_msg(RTAPI_MSG_ERR, "MX-PLUSE_GPIO: ERROR:
                    hal_malloc() failed\n");
    hal_exit(comp_id);
    return -1;
    }

    /* config addr and direction */
    for (i=0; i<8;i++){

        port_data_array->pin_addr[i] = pin_addr[i];
        port_data_array->pin_mask[i] = pin_mask[i];
    }


    /* export all vars */
    retval = export_port(port_data_array);
    if (retval != 0)
    {
        rtapi_print_msg(RTAPI_MSG_ERR, "MX-PLUSE_GPIO: ERROR: port
                        var export failed\n");
        hal_exit(comp_id);
        return retval;
    }

    return 0;
}


static int export_port(viagpio_t * port)
{
    int retval, msg;

    /* This function exports a lot of stuff, which results in a lot
    of logging if msg_level is at INFO or ALL. So we save the
    current value of msg_level and restore it later.   If you
    actually need to log this function's actions, change the second
    line below      */
    msg = rtapi_get_msg_level();
    rtapi_set_msg_level(RTAPI_MSG_WARN);

    retval = 0;

        // declare input pins (1 - 4)
        retval += export_input_pin(1, port->data_in, 0);
        retval += export_input_pin(2, port->data_in, 1);
        retval += export_input_pin(3, port->data_in, 2);
        retval += export_input_pin(4, port->data_in, 3);
```

```
        // declare output pins (5 - 8)
        retval += export_output_pin(5, port->data_out,
                  port->data_inv, 0);
        retval += export_output_pin(6, port->data_out,
                  port->data_inv, 1);
        retval += export_output_pin(7, port->data_out,
                  port->data_inv, 2);
        retval += export_output_pin(8, port->data_out,
                  port->data_inv, 3);

    port->write_time = 0;
    // restore saved message level
    rtapi_set_msg_level(msg);
    return retval;
}


static int export_input_pin(int pin, hal_bit_t ** base, int n)
{
    int retval;

    /* export write only HAL pin for the input bit */
    retval = hal_pin_bit_newf(HAL_OUT, base + (2 * n), comp_id,
            "viagpio.pin-%02d-in",pin);
    if (retval != 0)
    return retval;

    /* export another write only HAL pin for the same bit inverted
*/
    retval = hal_pin_bit_newf(HAL_OUT, base + (2 * n) + 1, comp_id,
            "viagpio.pin-%02d-in-not", pin);
    return retval;
}


static int export_output_pin(int pin, hal_bit_t ** dbase, hal_bit_t
* pbase, int n)
{
    int retval;

    // export read only HAL pin for output data
    retval = hal_pin_bit_newf(HAL_IN, dbase + n, comp_id,
            "viagpio.pin-%02d-out", pin);
    if (retval != 0) return retval;

    // export parameter for polarity
    retval = hal_param_bit_newf(HAL_RW, pbase + n, comp_id,
            "viagpio.pin-%02d-out-invert", pin);
    if (retval != 0) return retval;

    return retval;
}
```

Anexo E: Archivo de Configuración .HAL para el uso de GPIO

```
loadrt trivkins
loadrt [EMCMOT]EMCMOT base_period_nsec=[EMCMOT]BASE_PERIOD
servo_period_nsec=[EMCMOT]SERVO_PERIOD num_joints=[TRAJ]AXES
loadrt hal_viagpio
loadrt stepgen step_type=0,0,0


addf viagpio.read base-thread
addf stepgen.make-pulses base-thread
addf viagpio.write base-thread

addf stepgen.capture-position servo-thread
addf motion-command-handler servo-thread
addf motion-controller servo-thread
addf stepgen.update-freq servo-thread
net spindle-cmd <= motion.spindle-speed-out
net spindle-cw <= motion.spindle-forward
net spindle-ccw <= motion.spindle-reverse

net xdir => viagpio.pin-05-out
net xstep => viagpio.pin-06-out




setp stepgen.0.position-scale [AXIS_0]SCALE
setp stepgen.0.steplen 1
setp stepgen.0.stepspace 0
setp stepgen.0.dirhold 31732
setp stepgen.0.dirsetup 31732
setp stepgen.0.maxaccel [AXIS_0]STEPGEN_MAXACCEL
net xpos-cmd axis.0.motor-pos-cmd => stepgen.0.position-cmd
net xpos-fb stepgen.0.position-fb => axis.0.motor-pos-fb
net xstep <= stepgen.0.step
net xdir <= stepgen.0.dir
net xenable axis.0.amp-enable-out => stepgen.0.enable
net min-x => axis.0.neg-lim-sw-in
net max-x => axis.0.pos-lim-sw-in

setp stepgen.1.position-scale [AXIS_1]SCALE
setp stepgen.1.steplen 1
setp stepgen.1.stepspace 0
setp stepgen.1.dirhold 31732
setp stepgen.1.dirsetup 31732
setp stepgen.1.maxaccel [AXIS_1]STEPGEN_MAXACCEL
net ypos-cmd axis.1.motor-pos-cmd => stepgen.1.position-cmd
net ypos-fb stepgen.1.position-fb => axis.1.motor-pos-fb
net ystep <= stepgen.1.step
net ydir <= stepgen.1.dir
net yenable axis.1.amp-enable-out => stepgen.1.enable
```

```
net min-y => axis.1.neg-lim-sw-in
net max-y => axis.1.pos-lim-sw-in

setp stepgen.2.position-scale [AXIS_2]SCALE
setp stepgen.2.steplen 1
setp stepgen.2.stepspace 0
setp stepgen.2.dirhold 31732
setp stepgen.2.dirsetup 31732
setp stepgen.2.maxaccel [AXIS_2]STEPGEN_MAXACCEL
net zpos-cmd axis.2.motor-pos-cmd => stepgen.2.position-cmd
net zpos-fb stepgen.2.position-fb => axis.2.motor-pos-fb
net zstep <= stepgen.2.step
net zdir <= stepgen.2.dir
net zenable axis.2.amp-enable-out => stepgen.2.enable

net estop-out <= iocontrol.0.user-enable-out
net estop-out => iocontrol.0.emc-enable-in

loadusr -W hal_manualtoolchange
net tool-change iocontrol.0.tool-change =>
hal_manualtoolchange.change
net tool-changed iocontrol.0.tool-changed <=
hal_manualtoolchange.changed
net tool-number iocontrol.0.tool-prep-number =>
hal_manualtoolchange.number
net tool-prepare-loopback iocontrol.0.tool-prepare =>
iocontrol.0.tool-prepared
```

GLOSARIO DE TÉRMINOS

Acabado: Etapa de manufactura en la que se desea obtener la mayor calidad estética en la superficie del producto elaborado.

Código G: Lenguaje de programación más utilizado en el Control Numérico para el manejo de máquinas-herramienta.

Desbaste: Etapa de manufactura en la que se remueve la mayor cantidad de material posible y las partes superficiales y más duras.

Feedrate: Velocidad de avance entre la herramienta y la pieza de trabajo

.

Kernel: Núcleo, es la parte más importante de un Sistema Operativo y se encarga de proveer los servicios más básicos del sistema.

Makefile: Archivo que contiene la secuencia de instrucciones que se ejecutan para compilar al ejecutar el comando make.

Spindle Speed: Velocidad de rotación de la herramienta de corte en Revoluciones por minuto (RPM).

Toolpath: Trayectoria de movimiento que seguirá la herramienta, está escrito en código G.

BIBLIOGRAFÍA

[1] Miller, R. y Miller, M. R. , Audel Automated Machines and Toolmaking, Wiley, 2004.

[2] «The CNC Process,» CNC 4 everyone, 15 Febrero 2012. [En línea]. http://www.cnc4everyone.com/beginning-with-cnc/the-cnc-process/, fecha de consulta Diciembre 2014.

[3] Bowman, M. , CNC Milling in the Workshop, Crowood, 2013.

[4] Radhakrishnan, P. , Subramanyan, S. y Raju, V. , CAD/CAM/CIM, New Age International Pvt. Ltd., Publishers, 2008.

[5] Waters, T. F. , Fundamentals of Manufacturing For Engineers, Taylor and Francis, 2002.

[6] Overby, A. , CNC Machining Handbook, McGraw-Hill/TAB Electronics, 2010.

[7] «How do CNC's Work?,» CNC 4 everyone, 15 Febrero 2012. [En línea].

http://www.cnc4everyone.com/beginning-with-cnc/how-do-cncs-work/, fecha de consulta Diciembre 2014.

[8] Mattson, M. , CNC Programming: Principles and Applications, Cengage Learning, 2009.

[9] «CNC Machine Overview and Computer Numerical Control History,» CNCCookbook, [En línea]. http://www.cnccookbook.com/CCCNCMachine.htm, fecha de consulta Diciembre 2014.

[10] «The Different Types of CNC Machines,» CNC 4 everyone, 16 Febrero 2012. [En línea]. http://www.cnc4everyone.com/cnc-machines/the-different-types-of-cnc-machines/, fecha de consulta Diciembre 2014.

[11] «CAD and CAM,» CNC 4 everyone, 25 Febrero 2012. [En línea]. http://www.cnc4everyone.com/software/cad-and-cam/, fecha de consulta Diciembre 2014.

[12] «PyCAM,» 13 Junio 2011. [En línea]. http://pycam.sourceforge.net/ , fecha de consulta Enero 2015.

[13] «LinuxCNC,» 6 Abril 2014. [En línea]. http://www.linuxcnc.org/, fecha de consulta Enero 2015.

[14] «EMC internals,» 11 Noviembre 2007. [En línea]. http://www.linuxcnc.org/index.php/english/component/content/article/2-technical-articles/42-emc-internals, fecha de consulta Enero 2015.

[15] «PCM-9562,» Advantech, [En línea]. http://www.advantech.com/products/46DBCE98-D20D-4EEF-9047-D77D093B98F0/PCM-9562/mod_B2EDAB23-DFF4-4A72-9937-0EF466C37CFE.aspx, fecha de consulta Febrero 2015.

[16] «EPIA-P830,» VIA, [En línea]. http://www.viaembedded.com/en/boards/pico-itx/epia-p830/, fecha de consulta Febrero 2015.

[17] «Prusa Mendel (iteration 2),» RepRap, [En línea]. http://reprap.org/wiki/Prusa_Mendel, fecha de consulta Febrero 2015.