Utilización del Arduino para controlar motores de paso por medio del flasheo de GRBL

Alfredo Ricci, Juan Andrés Urrea

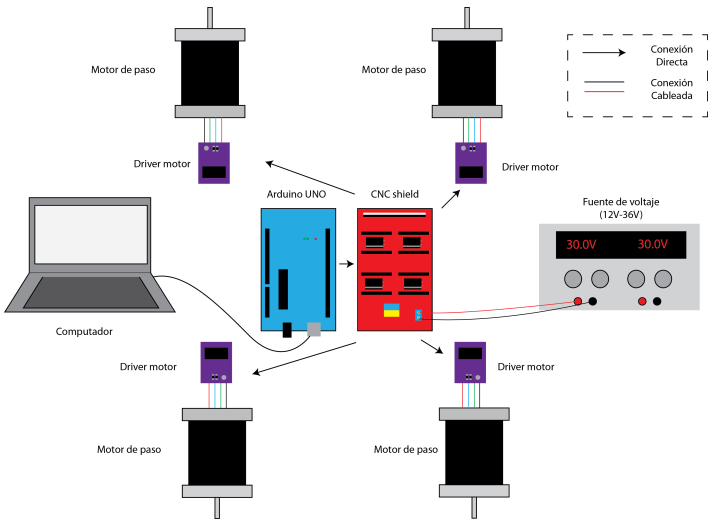
Resumen- Este informe busca mostrar la totalidad del procedimiento realizado para configurar un computador y un arduino como controladores para motores de paso. Se centra en la instalación del software necesario tanto en su utilización una vez esto se completa, al igual que la posterior creación y utilización de una interfaz gráfica en Matlab como herramienta para el usuario. De esta forma, explica el procedimiento de instalación y configuración, sirviendo a la vez como un tutorial para repetir este procedimiento.

1 Introducción

Dada la necesidad de tener un conocimiento previo del manejo tanto manual como automatizado de los motores utilizados para el proyecto **Mechanical Scanner**, se ideó la estrategia de reproducir esta situación a pequeña escala, usando motores de paso para luego conectarlos a un Arduino flasheado con GRBL y aprender, desde cero, los procesos necesarios para transmitir órdenes de trayectorias desde un computador hasta los motores de paso utilizados. De esta forma, la construcción exitosa de este montaje y la capacidad de controlar detalladamente estos dispositivos por medio de un manejo apropiado del software permitirá obtener este conocimiento previo a la llegada y utilización del **Mechanical Scanner**.

2 Montaje Utilizado

A continuación se presenta el diagrama que describe el montaje de conexión realizado entre el arduino, el cnc shield, los drivers, motores y computador como base de control.



3 Manejo de Software

3.1 Instalación de Software Utilizado

Antes de la explicación detallada acerca del proceso de instalación, se enuncian a continuación los diferentes programas y software utilizados:

- **GRBL**
  - Versión: 0.9

  GRBL es un controlador de movimiento disponible para funcionar con Arduino, escrito enteramente en *C*, que sirve como alternativa al control por puerto paralelo. Este software permite utilizar archivos escritos en *GCode* para diseñar trayectorias a seguir, de manera que estos se convierten en instrucciones para el Arduino, por medio de la instalación de GRBL, que luego son enviadas a la tarjeta controladora CNC.

- **Arduino UNO**
  - Versión 1.6.5

  Software que funciona como interfaz para controlar el dispositivo arduino desde un computador. Puede instalarse directamente desde la página web de Arduino. Software programado en Java, permite la creación de código que luego es subido directamente al dispositivo Arduino.

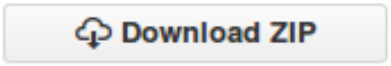
Instalación del Software Requerido

Una cez se sabe cual es el software requerido, se presenta ahora el proceso detallado de instalación de estos. Cabe notar que el proceso mostrado a continuación se realizó tanto en Linux como en Mac. Esto con la finalidad de utilizar los procedimientos más sencillos que ofrece cada sistema operativo. Para este caso, el flasheo de GRBL al Arduino se realizó desde Linux, mientras que la construcción y envío de GCode al Arduino se realizó desde Mac. Estas diferencias se enunciarán en cada paso.

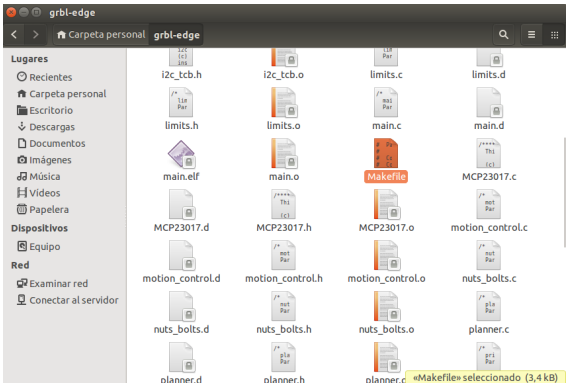
1. **Revisión de Librerías de Instalación Linux**
   Cuando se trabaja con Linux, especificamente la distribución Ubuntu 14.04 "Trusty", se debe realizar una verificación de librerías que permiten la instalación correcta de paquetes. Dicho proceso se puede realizar con los siguientes comandos de instalación en la terminal. En caso de que ya se encuentren actualizadas, en la terminal esto será notificado. En caso de que no, se instalarán.

```
sudo apt-get install ruby
sudo apt-get install avrdude
```

2. **Descargar GRBL y Flashearlo**
Ahora, es necesario descargar todo lo relacionado a GRBL. Esto se puede realizar desde GitHub, en el enlace [1]. Para hacer esto, se recomienda, si es que es la primera vez que se trabaja con esta herramienta, seleccionar la opción **Download Zip** en la parte derecha de la página.



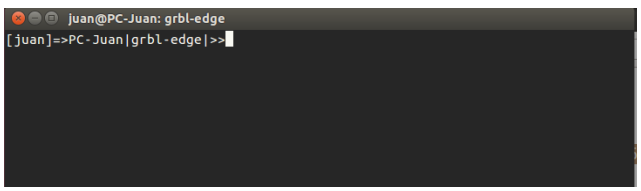
El archivo zip descargado debe luego ser descomprimido en una carpeta cuya localización resulte fácil de recordar. Debido a como está en GitHub, el archivo descargado se llamará **grbl-edge**. Para continuar este procedimiento, se abre ahora la carpeta recién descargada, buscando el archivo llamada **Makefile**, el cual debe ser después abierto con un editor de texto de preferencia personal, en este caso **Gedit**, editor de texto predeterminado de Ubuntu.



Una vez abierto este archivo, se busca la línea que empieza por la palabra *PROGRAMMER*, para reemplazarla por lo siguiente:

```
PROGRAMMER = -c stk500v1 -P /dev/
    ttyACM0 -b 115200
```

El archivo luego debe guardarse para actualizar y aplicar los cambios realizados. Una vez se hace esto, se debe abrir una terminal nueva. Esto para aprovechar las ventajas de control que esta aplicación ofrece. Desde esta se debe acceder a la carpeta **grbl-edge** creada anteriormente. Aunque la apariencia de la terminal depende sustancialmente de la configuración en cada equipo, para este caso específico se obtiene la siguiente ventana:



Una vez dentro de este entorno, se utiliza las siguientes líneas de código, para inicializar el archivo **make** dentro de la carpeta.

```
sudo make clean
sudo make
```

Realizar esto causará que aparezcan varias notificaciones en la terminal, las cuales sirven únicamente para informar de los procesos completados. Para finalizar este procedimiento, solo resta flashear GRBL al Arduino. Para hacer esto, se debe conectar el Arduino al computador en uso. Luego de hacer esto, se ejecuta en la terminal la siguiente línea de código.

```
sudo make flash
```

Esta línea instalará GRBL en el arduino, tras de lo cual se puede desconectar del computador y usar como controlador de movimiento de ahora en adelante.

4 Construcción de la Interfaz Gráfica en Matlab

Una vez se han instalado los programas necesarios para la realización del flasheo y este mismo se ha realizado, resulta ahora necesario la construcción de una interfaz gráfica y funcional desde Matlab que permita al usuario la creación de barridos de forma manual, el seleccionamiento de barridos predeterminados y en general el control del movimiento por medio de parámetros, a la vez que el recibimiento de una retroalimentación de posiciones constante. Con este objetivo se comienza entonces este procedimiento, empezando con la utilización de la función **GUIDE** de Matlab, ejecutando el siguiente comando en la consola:

```
>>guide
```

Este comando abre herramienta de creación de interfaz gráfica de Matlab, la cual comienza con el plano básico donde es posible insertar distintos tipos de elementos como botones, cuadros de texto editables, menus de despliegue, etc...A continuación se presentan las ventanas emergentes que dan comienzo a la creación de la interfaz gráfica.

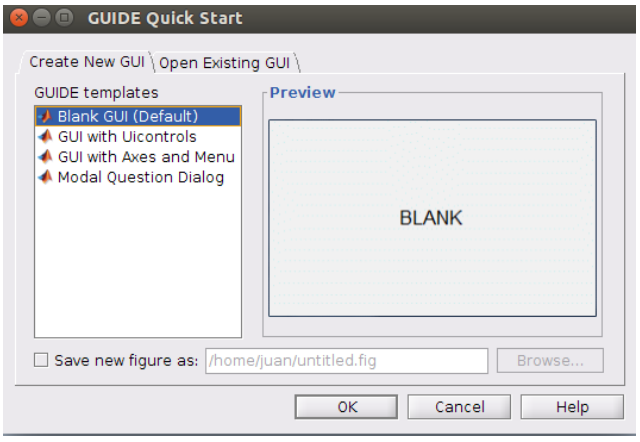


Figure 1: Selección de Formato

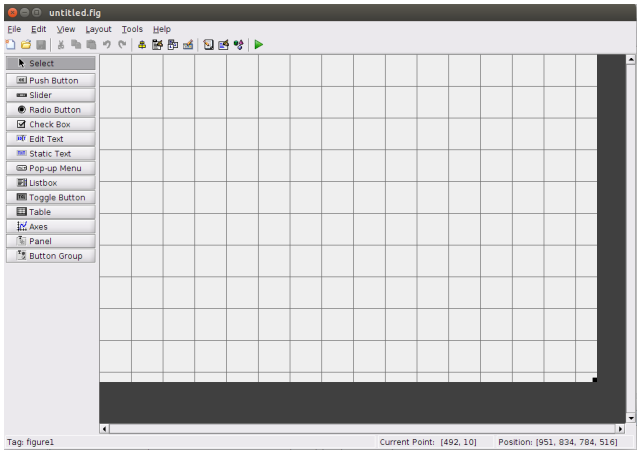


Figure 2: Ventana GUI nueva

En la figura se observa la selección de plantilla que puede utilizarse para la construcción de una nueva interfaz. Para este caso se selecciona la opción *Blank GUI (Default)*, puesto que no se busca partir de una plantilla predeterminada. Es posible también selecionar la casilla **Save new figure as**, de manera que se guarde desde ese momento la nueva figura creada. De hacer esto, se guardará en la carpeta deseada como archivo **.fig**, el cual puede ser modificado desde la opción *guide* nuevamente cuando sea necesario.

Una vez se ha dado la opción **OK** y se ha elegido la plantilla en blanco, se despliega la ventana mostrada en la figura , donde es posible comenzar la construcción de la interfaz deseada. Para trabajar específicamente con el modelo planeado para esta interfaz, mostrado a continuación, se exploran únicamente las opciones y herramientas que sirven para la creación de esta, encontrándose un tutorial más detallado de las demás opciones en [1]. **MONTAR FOTO** A partir del modelo mostrado, se procede, inicalmente, a construir la estética de la interfaz, proceso que consiste en el solo posicionamiento de cada elemento dentro de la interfaz, sin trabajar aún en las funciones que cada uno cumple. Esto se hace seleccionando cada item mostrado a la izquierda en la figura y llevándolo hasta la posición deseada. Esto se hace entonces con los elementos correspondientes a *Push Button*, *Static Text*, *Edit Text*, *Pop-Up Menu* y *Check Box*. A continuación se enuncian brevemente las características de cada uno:

- **Push Botton:** Corresponde a un botón en la interfaz, de manera que al pulsarse se realiza algún procedimiento correspondiente.

- **Static Text:** Corresponde a un cuadro de texto que puede ser editado por funciones de la interfaz mas no por el usuario, usado para mostrar resultados esencialmente de respuesta.

- **Edit Text:** Corresponde a un cuadro de texto que puede ser editado por el usuario, de manera que la interfaz puede recibir como parámetro para funciones el valor ingresado aquí.

- **Pop-Up Menu**: Corresponde a un menu desplegable que ofrece opciones que han sido definidas previamente en la construcción de la interfaz, de manera que la selección de cada opción conlleva a la realización de un procedimiento distinto.

- **Check Box:** Corresponde a una casilla de selección con la posibilidad de ser marcada o no. Este estado definirá entonces una condición para la realización de procesos internos de la interfaz.

Una vez se ha posicionado cada elemento, es posible ahora editar el nombre con el que será identificado dentro de las funciones internas de la interfaz, a la vez que el texto que se ve sobre el en la estética de la interfaz. Para realizar este procedimiento, se da click derecho sobre cada elemento y se selecciona la opción **Property Inspector**, donde se muestran entonces las distintas propiedades editables disponibles para cada tipo de elemento. Para el procedimiento aquí realizado, resulta únicamente necsario la edición de los campos **Tag** y **String**, los cuales representan el nombre bajo el cual se identifica el elemento dentro de las funciones de la interfaz y el texto que se ve sobre el elemento en la interfaz, respectivamente. El resultado de este procedimiento se muestra a continuación:

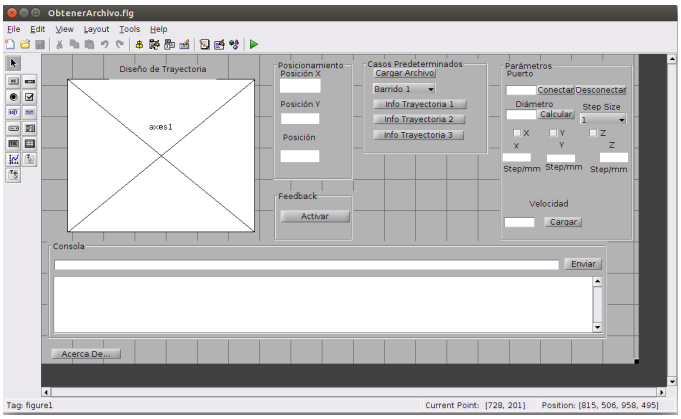


Figure 3: Interfaz construida en GUI

Cabe notar la utilización del elemento **Panel**, el cual sirve únicamente como herramienta de estética para crear divisiones dentro de la interfaz para crear orden.La creación del título correspondiente a cada uno se realiza de igual forma en la ventana correspondiente de *Property Inspector*. Con este procedimiento se ha finalizado ya la etapa estética de la construcción de la interfaz, por lo que se procede ahora a la construcción de las funciones internas que dan a cada elemento utilizado un funcionamiento o relación con otros según se necesita en la aplicación. Para comenzar a construir dichas funciones, se establecen entonces las diferentes funciones que se necesitan de primera mano por parte del usuario, al igual que aquellas que deben crearse internamente como consecuencias de estas. Se enuncian a continuación las funciones generales que la interfaz debe cumplir para dar al usuario control sobre el movimiento de los motores, al igual que información acerca de la posición de estos en todo momento.

Requerimientos Funcionales

A continuación se enuncian los requerimientos funcionales generales que debe cumplir la interfaz como medio de comunicación entre el usuario y el montaje de los motores.

(a) Permitir al usuario mandar código G que defina el movimiento deseado de manera manual.

(b) Permitir al usuario enviar una trayectoria predeterminada a realizar, ya sea de un archivo de código G escrito previamente o una trayectoria construida por la interfaz por medio de parámetros de entrada.

(c) Proveer al usuario una retroalimentación constante tanto como verificación de lo que fue enviado por este tanto de la posición actual dada por el montaje, todo por medio de una consola en tiempo real.

(d) Permitir al usuario modificar parámetros del montaje para los cálculos internos de GRBL, como lo son el diámetro de las ruedas utilizadas, el $\frac{Step}{mm}$ en cada dirección, etc...

(e) Permitir al usuario definir manualmente el puerto serial donde se trabajará la conexión, de manera que se pueda establecer la relación entre la interfaz de Matlab y el montaje.

Generalidades acerca del envío de comandos a GRBL Versión 0.9

Cada versión disponible actualmente de GRBL cuenta con una serie de comandos específicamente designados para que, al ser enviados por puerto serial al montaje de CNC, devuelva información específica de este o modifique un parámetro de su movimiento. La lista entera se puede encontrar en el repositorio oficial de GitHub[2], sin embargo a continuación se enuncian los que son utilizados en la construcción de la interfaz de usuario y el código que define sus funciones:

- $100
  Al introducir este comando por puerto serial y enviarlo a GRBL igualado a un valor numérico, este valor se tomará como el $\frac{Step}{mm}$ en la dirección X. Un ejemplo es:

  ```
  >>$100=25.6
  ```

  Donde el $\frac{Step}{mm}$ tomará el valor de 25.6.

- $101
  Al introducir este comando por puerto serial y enviarlo a GRBL igualado a un valor numérico, este valor se tomará como el $\frac{Step}{mm}$ en la dirección Y. Un ejemplo puede ser el mismo que el mostrado para la dirección X, escribiendo ahora $101.

- $102
  Al introducir este comando por puerto serial y enviarlo a GRBL igualado a una valor numérico, este valor se tomará como el $\frac{Step}{mm}$ en la dirección Z. El ejemplo es igual a los mostrados anteriormente.

Teniendo definidos estos requerimientos, se procede entonces a construir el código dentro del archivo **.m** que se creó simultáneamente a la creación del archivo **.fig**. Para hacer esto, se comienza entonces con todo lo relacionado a la selección de archivos externos o trayectorias predeterminadas. Esto corresponde al panel denominado *Casos Predeterminados*. Se muestra a continuación el código detallado y documentado para cada aspecto necesario de este grupo de funciones:

- **Cargar Archivo**
  Esta función permite al usuario cargar al montaje un archivo escrito en código G de manera que cada línea se envíe por consola al montaje. Para lograr hacer esto, el archivo seleccionado se importa y su contenido se almacena tempiralmente línea por línea en una variable, siendo este luego enviado al puerto serial.

```
function Cargar_Callback(hObject,
    eventdata, handles)
%Se crea la ventana para cargar
    archivos de tipo .g.
[Filename Path]=uigetfile({'*.g'},'
    Abrir archivo');
%Se establecen los casos para actuar
    segun lo seleccionado.
if isequal(Filename,0)
    return
else
    %Se recibe el archivo y su
        contenido es leido, almacenado
        y enviado.
end
```

- **Cargar Trayectoria Predeterminada**
  Esta función permite al usuario desplegar un menú en lista donde se muestran las distintas trayectorias predeterminadas que pueden ser utilizadas para el montaje. De seleccionar una de la lista, en el eje denominado *Axes*, mostrado en la figura , se debe mostrar la imagen de esta trayectoria, a la vez que se despliega una ventana emergente para ingresar los parámetros geométricos que definen cada trayectoria.
  En Proceso: Debido a que la totalidad de esta función no se ha finalizado, solo se muestran a continuación las acciones parciales que realiza hasta ahora. estas son la aparición de una foto que identifica cada trayectoria. Apenas se tengan los scripts que definen cada trayectoria según parámetros, se actualizará el código correspondiente a continuación.

```
function Predet_Callback(hObject,
    eventdata, handles)
%Definir los distintos valores que
    pueden ser adoptados en la
    seleccion.
fun = get(hObject,'Value');
    switch fun
    %Definir cada caso y la accion que
        se toma en cada uno.
```

```
        case 1
        %Se muestra en el eje de
            imagenes la imagen
            correspondiente a cada
            trayectoria.
            a = imread('A.png');
            imshow(a);
            %Se despliega la ventana
                emergente para ingresar
                 los parametros de cada
                 trayectoria.
            x=inputdlg({'Parametro A','
                Parametro B'},'
                Parametros');
            %x sera una celda de dos
                elementos.
            params=cell2mat(x);
            %params es un arreglo con
                estos valores.
                Posteriormente sera
                posible manipularlos
                para la creacion de la
                trayectoria.
        case 2
            b = imread('B.jpg');
            imshow(b);
        case 3
            c = imread('C.png');
            imshow(c);

    end
guidata(hObject,handles);
```

- **Dar Información acerca de cada Trayectoria**
  Esta función permite al usuario obtener información general acerca de cada una de las trayectorias predeterminadas disponibles en la forma de mensajes informativos en ventanas emergentes, activadas por el botón correspondiente. Esta herramienta viene definida entonces para tres botones distintos; **Info Trayectoria 1**, **Info Trayectoria 2** y **Info Trayectoria 3**. Por le momento, dado que no se han definido estas trayectorias, se muestra un mensaje genérico para la trayectoria 1, mientras que permanece siendo posible editar los tres emnsajes una vez se construyan correctamente las trayectorias.

```
function Info1_Callback(hObject,
    eventdata, handles)
msgbox('Esta trayectoria realiza un
    barrido dado por la geometria
    mostrada en la figura. Para
    construirlo, el usuario debe dar
    los parametros requeridos.','
    Trayectoria 1');
%Info de la trayectoria 1.
```

Ahora se continua con el panel relacionado a la modificación de los parámetros de conexión y configuraciones de los motores utilizados, como lo son la selección del puerto serial donde se encuentra conectado el Arduino, los $\frac{Step}{mm}$ en cada dirección, la velocidad de los motores, etc... Para lograr todas estas funcionalidades, se implementó el código mostrado y documentado a continuación.

- **Selección de Puerto Serial**
  Esta función permite al usuario seleccionar el puerto serial al cual quiere realizar la conexión y hacerla, haciendo uso del botón **Conectar**. De manera similar, permite cerrar dicha conexión con el botón **Desconectar**.

```
function Connect_Callback(hObject,
    eventdata, handles)
global s; %Se declara s como una
    variable global para que todas las
    funciones puedan acceder a esta si
    la necesitan.
puerto=handles.Port; %Se establece la
    variable local puerto como lo que
    el usuario escribe en el campo de
    Puerto.
s=serial(puerto); %Declarar puerto
    serial.
fopen(s); %Abrir la conexion al puerto
    serial.
```

- **Desconectar**
  Esta función permite cerrar la conexión serial previamente establcida con el puerto seleccionado, haciendo referencia a la variable local que lo contiene.

```
function Disconnect_Callback(hObject,
    eventdata, handles)
global s; %Permite modificar la
    variable s.
fclose(s); %Cierra la conexion al
    puerto serial.
```

- **Calcular el $\frac{Step}{mm}$ en cada eje**
  Esta función permite determinar el $\frac{Step}{mm}$ en cada eje de dirección, basándose en parámetros de entrada como lo son el diámetro de las ruedas o poleas a utilizar y el *Step Size*, seleccionado de una lista desplegable, al igual que seleccionando la dirección que se desea trabajar. El código implementado para lograr esto se muestra documentado a continuación. Ya que este proceso requiere la utilización de varios campos de texto junto con botones, se muestran las distintas funciones separadas.

```
function checkbox1_Callback(hObject,
    eventdata, handles)
%Se define la funcion del checkbox de X
    .
global chuleadoX; Se torna la variable
    en global para poder referenciarla
    desde otras funciones.
%En caso de estar marcado, la variable
    gobal toma el valor 1, en caso
    contrario toma el valor 0.
```

```
if(get(hObject,'Value')== get(hObject,'
    Max'))
     chuleadoX=1;
else
     chuleadoX=0;
end

%Las funciones para los dos checkbox
    restantes tiene la misma estructura
    , variando unicamente la variable
    global que se modifica.

function checkbox2_Callback(hObject,
    eventdata, handles)
global chuleadoY;
if(get(hObject,'Value')==get(hObject,'
    Max'))
     chuleadoY = 1;
else
     chuleadoY=0;
end

function checkbox3_Callback(hObject,
    eventdata, handles)
global chuleadoZ;
if(get(hObject,'Value')==get(hObject,'
    Max'))
     chuleadoZ=1;
else
     chuleadoZ=0;
end

function Diameter_Callback(hObject,
    eventdata, handles)
%La funcion define como utilizar el
    parametro de entrada del valor de
    diametro.
Val=get(hObject,'String'); %Se recibe
    el parametro de tipo String.
NewVal=str2double(Val); %Se convierte
    el valor en un caracter numerico
    decimal.
handles.Diameter=NewVal; %Se guarda el
    valor de la variable.
guidata(hObject,handles); %Se guarda la
     variable en la informacion de la
    funcion.

function Pasos_Callback(hObject,
    eventdata, handles)
global Step; %Se declara la variable
    global para ser accesible por otras
     funciones.
fun=get(hObject,'Value')
%Se declara una variable local cuyo
    valor esta dado por las opciones
    del menu desplegable.
     switch fun
     %La variable global Step tomara el
         valor correspondiente al Step
         Size deseado por el usuario.
         case 1
            Step = 1;
         case 2
             Step = 1/2;
         case 3
             Step = 1/4;
         case 4
             Step = 1/8;
         case 5
             Step = 1/16;
         case 6
             Step = 1/32;
     end
     guidata(hObject, handles);

function Calcular_Callback(hObject,
    eventdata, handles)
%Se importan las variable globales
    definidas en funciones anteriores,
    que son necesarias para el calculo
    del Step/mm.
global Step;
global chuleadoX;
global chuleadoY;
global chuleadoZ;
global s;
global consola;

%Primero, se definen excepciones
    generales, de manera que sea
    necesario modificar de a una
    variable a la vez, por lo que habra
     un mensaje de error en caso de
    seleccionar mas de una en los
    checkbox, a la vez que existira un
    error si no se selecciona ninguna
    para modificar.
if((chuleadoX == 1) && (chuleadoY == 1)
    && (chuleadoZ == 1))
     errordlg('Se debe seleccionar solo
         una coordenada para modificar.'
         ,'Error de Parametros');
end
if(chuleadoX == 1 && chuleadoY ==1 &&
    chuleadoZ == 0)
     errordlg('Se debe seleccionar solo
         una coordenada para modificar.'
         ,'Error de Parametros');
end
if(chuleadoX == 0 && chuleadoY ==1 &&
    chuleadoZ == 1)
     errordlg('Se debe seleccionar solo
         una coordenada para modificar.'
         ,'Error de Parametros');
end
if(chuleadoX == 1 && chuleadoY ==0 &&
    chuleadoZ == 1)
```

```
    errordlg('Se debe seleccionar solo
        una coordenada para modificar.'
        ,'Error de Parametros');
end
if(chuleadoX == 0 && chuleadoY ==0 &&
    chuleadoZ == 0)
    errordlg('Se debe seleccionar una
        coordenada para modificar.','
        Error de Parametros');
end

%Cuando no se presenta ningun error, se
    procede a realizar el calculo para
    la variable seleccionada.
if(chuleadoX == 1 && chuleadoY==0 &&
    chuleadoZ==0)%Condiciones de los
    checkbox.
%Se calcula el step por mm y se muestra
    , para luego enviarlo al CNC.
    calculo = (200*(1/(Step))/(pi*
        handles.Diameter));
    set(handles.PasosX,'String',calculo
        );
    fprintf(s,strcat('$100=',num2str(
        calculo)));
    lectura=fscanf(s,'%s'); %Leer el
        feedback
    consola = strvcat(consola,strcat('
        >>>','$100=',num2str(calculo)),
        lectura);
    set(handles.Resultado,'String',
        consola);
end

%Este mismo procedimiento se repite
    para los casos en los cuales la
    coordenada Y o Z hayan sido
    seleccionadas.

if(chuleadoY==1 && chuleadoX==0 &&
    chuleadoZ==0)
    calculo = (200*(1/(Step))/(pi*
        handles.Diameter));
    set(handles.PasosY,'String',calculo
        );
    fprintf(s,strcat('$101=',num2str(
        calculo)));
    lectura=fscanf(s,'%s'); %Leer el
        feedback.
    consola = strvcat(consola,strcat('
        >>>','$101=',num2str(calculo)),
        lectura);
    set(handles.Resultado,'String',
        consola);
end

if(chuleadoZ==1 && chuleadoX==0 &&
    chuleadoY==0)
    calculo=(200*(1/(Step))/(pi*handles
        .Diameter));
    set(handles.PasosZ,'String',calculo
        );
    fprintf(s,strcat('$102=',num2str(
        calculo)));
    lectura=fscanf(s,'%s'); %Leer el
        serial en formato string.
    consola = strvcat(consola,strcat('
        >>>','$102=',num2str(calculo)),
        lectura);
    set(handles.Resultado,'String',
        consola);
end

function Speed_Callback(hObject,
    eventdata, handles)
%Esta funcion define como se recibe y
    almacena el parametro de velocidad.
Val=get(hObject,'String'); %Recibir el
    parametro de la velocidad como
    String.
handles.Speed=Val; %Guardar la variable
    temporal para ser referenciada por
    otras funciones.
%El resto de esta funcion sigue en
    construccion.
guidata(hObject,handles);
```

Ahora, se procede a construir todo lo correspondiente al Panel de **Consola**, el cual contiene tres elementos principales:

(a) La barra de texto editable donde se introducen los comandos de código G o de edición de parámetros de manera manual por el usuario.

(b) El botoón denominado **Enviar**, el cual envía por medio de puerto serial dichos comandos introducidos.

(c) La celda de texto no editable que va mostrando el feedback de confirmación que devuelve el CNC, al igual que muestra el historial de comandos enviados por el usuario.

El código específicamente implementado para dar funcionalidad a estos tres elementos se muestra documentado a continuación.

```
function Comandos_Callback(hObject,
    eventdata, handles)
Val=get(hObject,'String'); %Recibir el
    comando escrito en la celda de texto
    editable.
NewVal = char(Val); %Convertir a arreglo de
    tipo Char.
handles.Comandos=NewVal; %Guardar el
    contenido dentro de la informacion de
    la funcion.
guidata(hObject,handles);
```

```
function Enviar_Callback(hObject, eventdata
    , handles)
global s; %Importar la variable global que
    define el puerto serial.
global consola; %Importar la variable
    global que contiene el historial de
    comandos enviados.
input = handles.Comandos; %Importar arreglo
     de Char.
consola = strvcat(consola,strcat('>>',input
    )); %Actualizar la variable Consola
    con los comandos enviados.
set(handles.Resultado,'String',consola); %
    Mostrar en la celda de texto de
    feedback la nueva variable consola
    actualizada.
fprintf(s,input); %Enviar comando
    directamente por puerto serial.
```

A continuación, se procede entonces a la construcción de las herramientas que permiten a la interfaz recibir la retroalimentación dada por el CNC tras el envío de comando por parte del usuario, de manera que la consola muestre no solo los comandos utilizados previamente, sino también las respuestas del CNC a estos, a manera de confirmar su correcto recibimiento o enterarse de cualquier error que pudiese ocurrir en el envío. Para realizar este procedimiento, se realiza una lectura periódica del puerto serial y los resultados obtenidos se imprimen automáticamente en la consola, utilizando para esto el código mostrado a continuación.

```
function feedbackButton_Callback(hObject,
    eventdata, handles)
global consola;
global s;
limite = 1000; %Tiempo en segundos durante
    el cual se realiza la lectura.
contador = 0; %Contadora.
borrar = 0; %Decidir cuando borrar.
while(contador < limite)
    lectura=fscanf(s,'%s'); %Leer el serial
        en formato string.
    consola = strvcat(consola,lectura); %
        Anadir a variable de consola.
    set(handles.Resultado,'String',consola)
        ; %Mostrar en consola de interfaz.
    contador = contador +1;
    pause(0.1); %El descanso.
end
```

La función definida y documentada anteriormente se ejecuta cuando el botón **Activar** del panel **Feedback** se oprime, de manera que paralelemanete a cualquier acción que se esté realizando, se realiza también una lectura constante del serial cada 0.1 segundos hasta completar 1000 lecturas consecutivas. Adicionalmente a esta función, se crea una código que complementa cada función que envía comandos al puerto serial del CNC, de manera que la respuesta de este a estos códigos es leída e impresa en la consola de la interfaz. Dicho complemento se muestra a continuación:

```
lectura=fscanf(s,'%s'); %Leer el feedback.
```

Esta línea de código se coloca justo despues de la línea que incluye el comando **fprintf()**, de manera que tras recibir el código enviado, el CNC envía una respuesta que la interfaz recibe y provee al usuario por medio de la consola.

5 Construcción de Trayectorias predeterminadas

Para el desplazamiento del dispositivo a través del área sobre la cual se desea trabajar, es necesario tener en cuenta los parametros de entrada con los cuales se carcteriza el área de barrido y la dimensión de los desplazamientos dentro de la misma. Para esto, se definen los siguientes parámetros:

- A= Ancho del área.
- h= Altura del área.
- S_h=Cantidad de pasos en el eje y
- S_A=Cantidad de pasos en el eje x
- S_x= Desplazamiento en la dirección de A ($\frac{A}{S_A}$).
- S_y= Desplazamiento en la dirección de h ($\frac{h}{S_h}$).
- v= Velocidad de Desplazamiento

En la siguiente figura se puede observar una representación típica del área de trabajo con los parametros previamente descritos.

Adicionalmente, se consideran los siguientes códigos códigos en G-code.

Comando	Descripción del comando
G21	Define las unidades en milímetros
G90	Define las coordenadas absolutas
G91	Define las coordenadas incrementales
G0	Movimiento rápido
G1	Movimiento a velocidad dada
f	Velocidad de desplazamiento
G92	Define la posición de origen absoluta

Para crear la trayectoria en matlab se inicia por definir una función con 5 parámetros y una salida. La salida es un vector de cadenas de caracteres que representan el código G que se desea enviar. Los parámetros de entrada son: A, h, S_h, S_A, v. Ahora, el primer paso es definir las variables S_x y S_y ya que estás funcionan como los parámetros del código G. La sintaxis correcta para el código de movimiento es la siguiente:

G00XxxYyyZzz

En donde xx, yy y zz representan la distancia que se desea mover en las respectivas coordenadas. De este modo:

$$xx = S_x \tag{1}$$
$$yy = S_y \tag{2}$$

De este modo, es necesario crear una función que dados los parámetros regrese el código necesario.

6 Bibliografía y Referencias

1. **Manual de Interfaz Gráfica de Usuario en Matlab** Parte 1. Barragán Guerreo, D. O. Archivo PDF.
2. **GRBL** Repositorio de GRBL en Github. Disponible en https://github.com/grbl/grbl.
3. **MathWorks** Documentación de Matlab. Disponible en www.mathworks.com.