LA UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA MECÁNICA

Farmbot simulator: Manual de desarrollador

Estudiantes
Víctor Alexander Murcia Vargas
201416659
Juan Felipe Palacios Sanchez
201616389

Asesor
Giacomo Barbieri, PhD

Índice

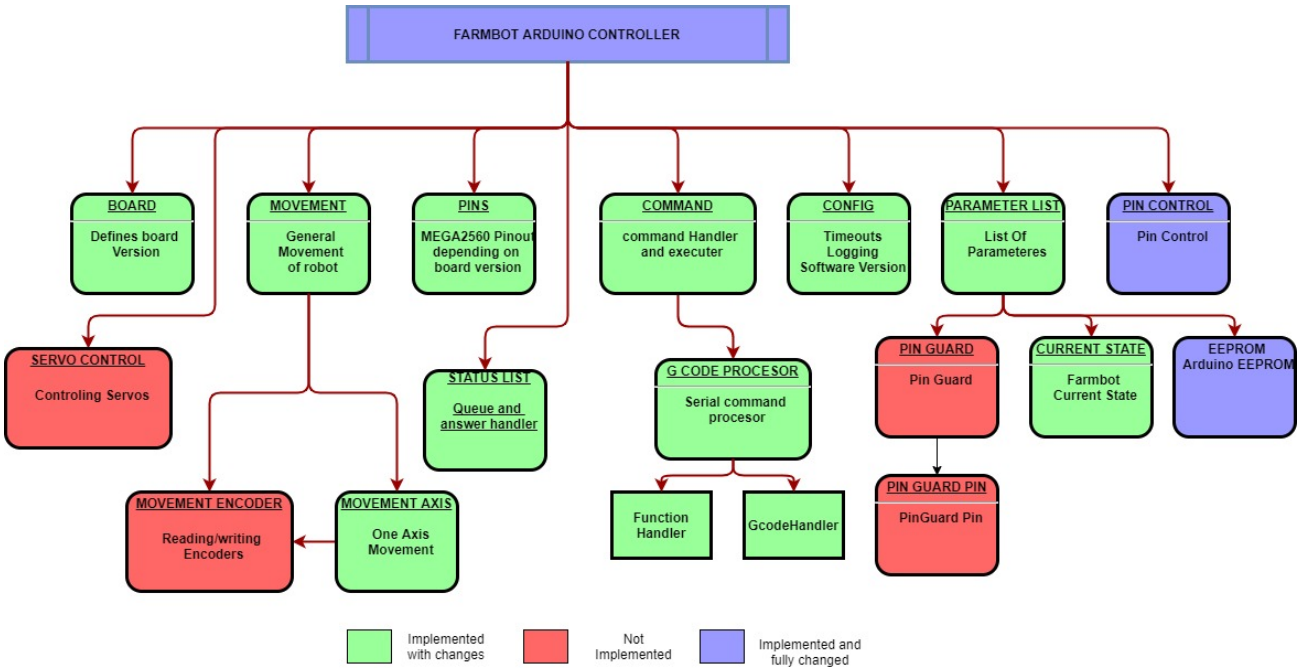
1. Introducción

El presente manual es una guía sobre como modificar el código escrito para el "Farmbot simulator.[en] C++/CLI. El código fue escrito basado en el Firmware original del farmbot en la versión 1.4. Inicialmente se presenta la arquitectura de software de la aplicación, se muestra como configurar los diferentes programas a utilizar para trabajar con el código.Enseguida,se describe cada una de las clases en detalle y por ultimo se presentan los posibles trabajos futuros

2. Arquitectura de Sotware

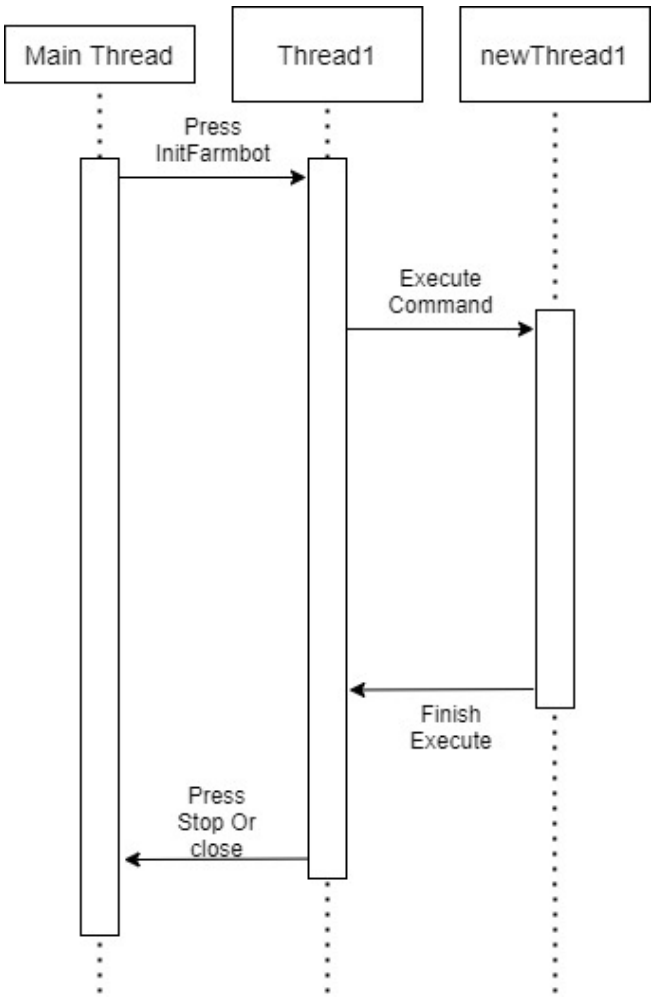
2.1. Diagrama de Clases

En la siguiente figura se muestra el diagrama de clases de esta aplicacion, este no es pertinentemente una jerarquia, debido a que la mayoria de los objetos del farmbot son declarados como variables globales.



2.2. Scheduling

En la siguiente figura se muestra el diagrama de hilos para la interfaz, esta solo cuenta con dos hilos, uno que se activa despues de iniciar el farmbo y otro que lo hace cada vez que se ejecuta un comando desde la consola serial.



3. Configuración del software requerido

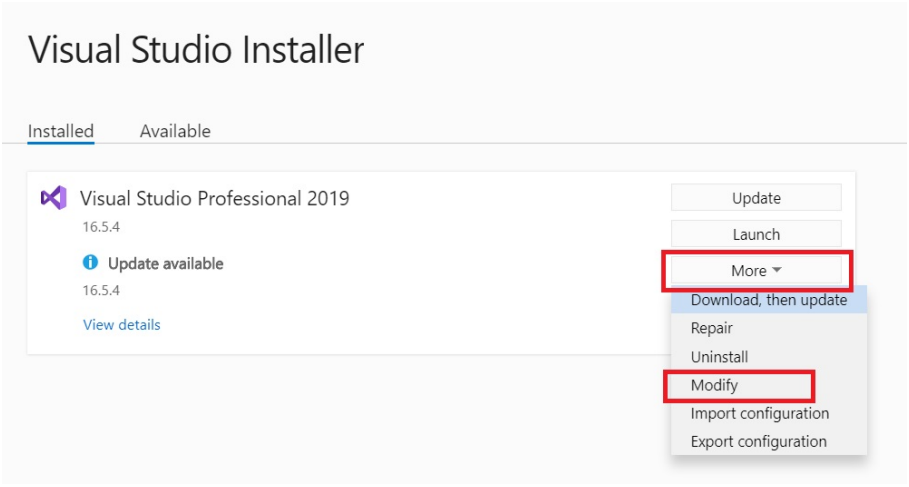
3.1. Instalacion de Visual Studio

Este código fue escrito usando visual Studio Professional V19, las instrucciones de descarga e instalación están en el siguiente link.

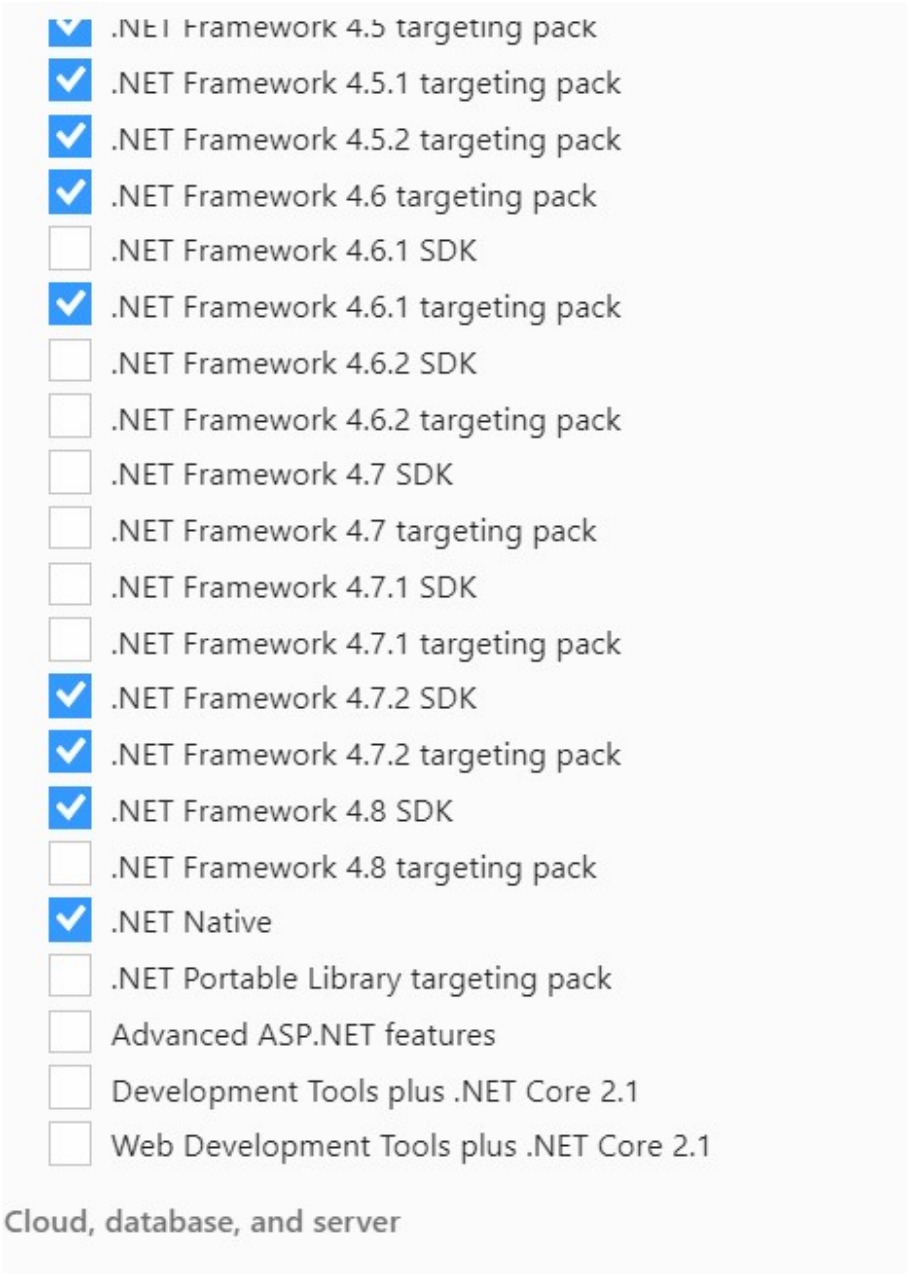
3.2. Instalación de Paquetes

Los paquetes que se muestran en la figura deben ser instalados antes de abrir el codigo. Para instalar los paquete:

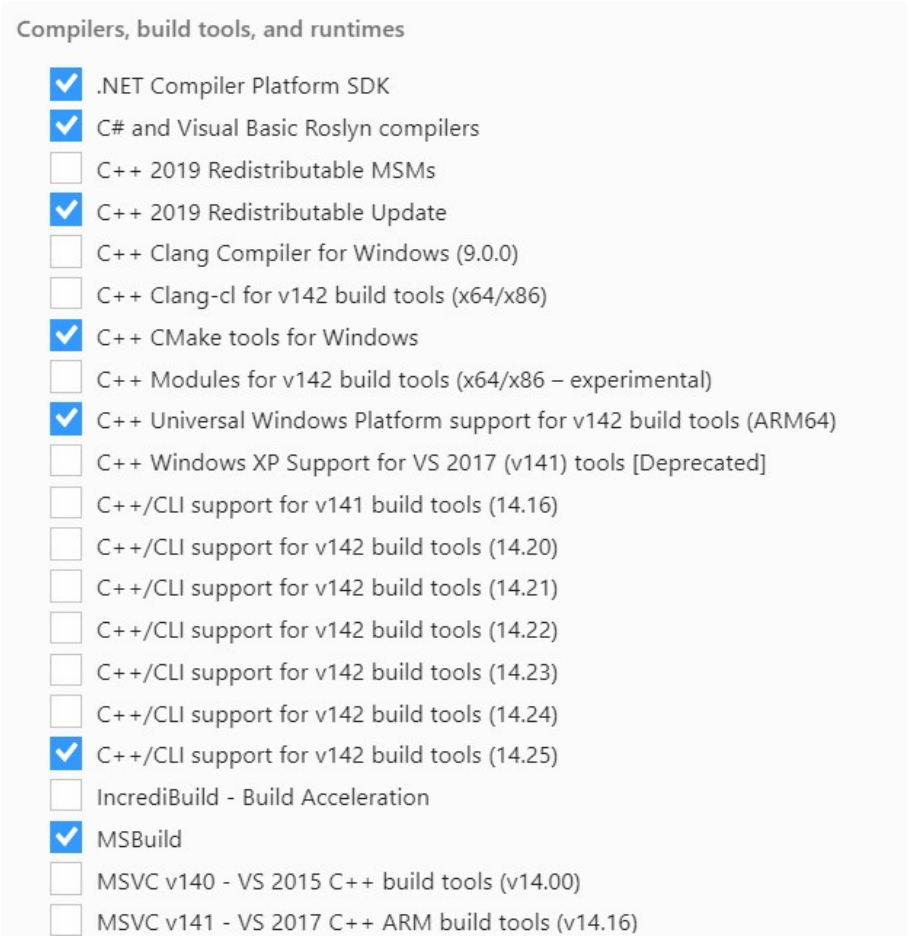
- Abra VisualStudio Installer
- En su versión del instalador seleccione More->Modify

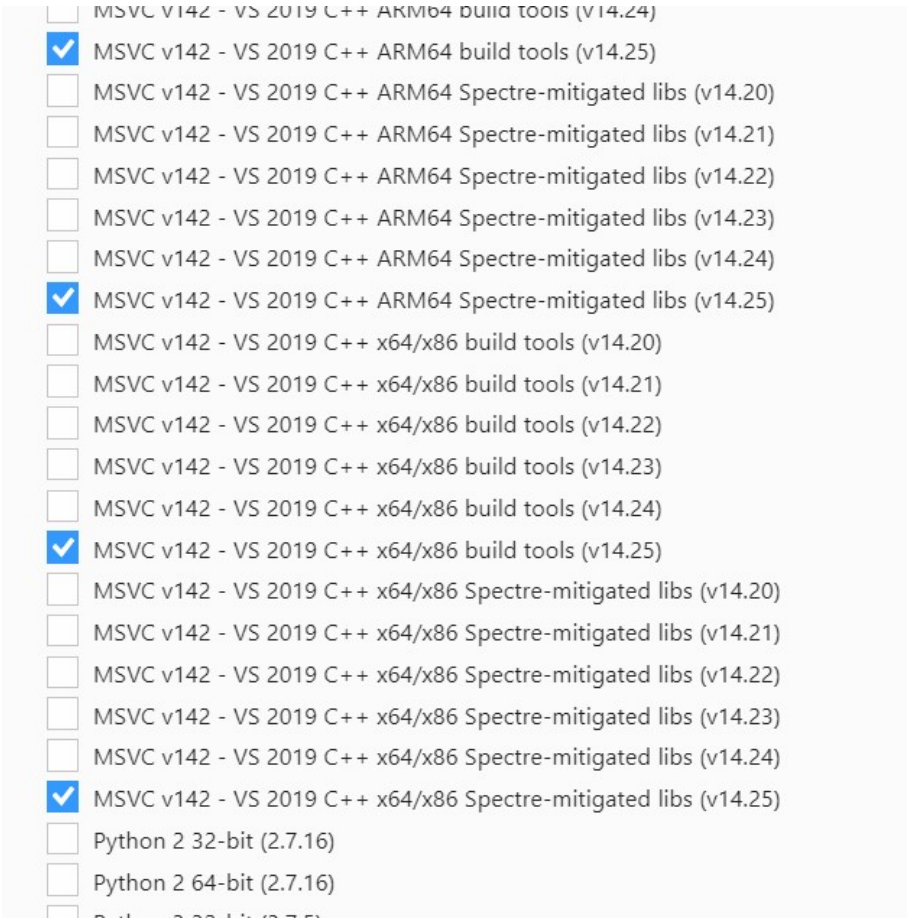


- Seleccione İndividual Components"
- Navegue al menú ".NETz verifique que concuerde con la figura



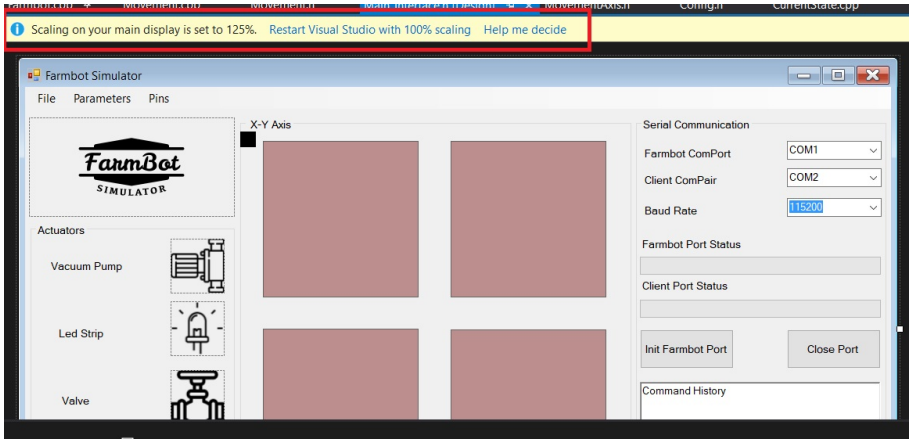
- Navegue al menú Çompilers, build tools, and runtimesz verifique que concuerde con las siguientes figuras


Compilers, build tools, and runtimes
.NET Compiler Platform SDK
C# and Visual Basic Roslyn compilers
C++ 2019 Redistributable MSMs
C++ 2019 Redistributable Update
C++ Clang Compiler for Windows (9.0.0)
C++ Clang-cl for v142 build tools (x64/x86)
C++ CMake tools for Windows
C++ Modules for v142 build tools (x64/x86 – experimental)
C++ Universal Windows Platform support for v142 build tools (ARM64)
C++ Windows XP Support for VS 2017 (v141) tools [Deprecated]
C++/CLI support for v141 build tools (14.16)
C++/CLI support for v142 build tools (14.20)
C++/CLI support for v142 build tools (14.21)
C++/CLI support for v142 build tools (14.22)
C++/CLI support for v142 build tools (14.23)
C++/CLI support for v142 build tools (14.24)
C++/CLI support for v142 build tools (14.25)
IncrediBuild - Build Acceleration
MSBuild
MSVC v140 - VS 2015 C++ build tools (v14.00)
MSVC v141 - VS 2017 C++ ARM build tools (v14.16)



3.3. Configuración del escalamiento

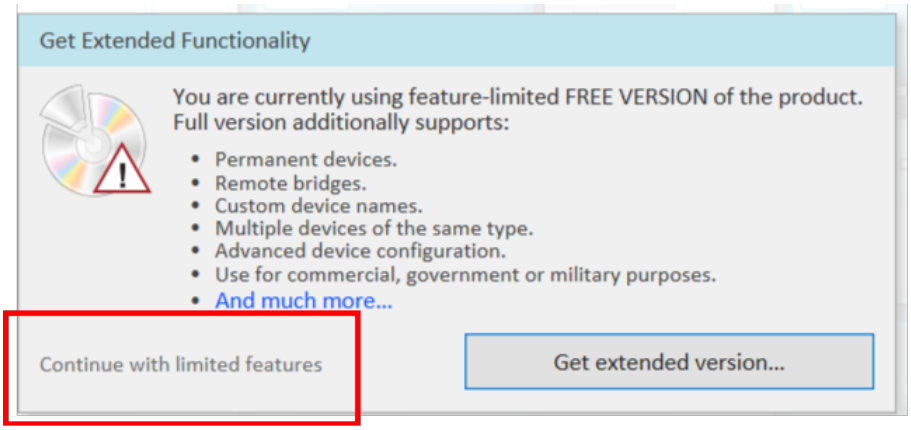
Una vez abierto el proyecto y teniendo en cuenta que visual Studio es una aplicación que funciona por DPI, al abrir el formulario en forma de diseño este aparecerá escalado mostrando la ventana que se muestra en la figura



3.4. Virtual Serial Ports Free

El instalador del programa se encuentra en la carpeta llamada "Virtual Serial", para instalarlo ejecute el archivo como administrador del sistema y siga los pasos. Si desea la ultima versión del programa la puede descargar del siguiente link.El programa corriendo se muestra en la figura.Para configurar los puertos seriales para esta aplicacion realice los siguientes pasos

- Abra el programa con permisos de administrador
- Haga click en Çontinue with limited features"



- Haga click en Çreate Local Bridge"

- Seleccione los nombres de los puertos, tenga en cuenta estos nombres para posteriormente usarlos en las aplicaciones.

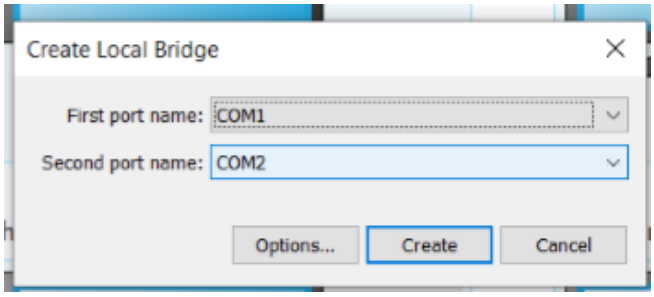


Figura 1:

- Presione click en Ok, tenga en cuenta que estos puertos existirán unicamente hasta que reinicie el sistema, es decir cada vez que lo reinicie deberá seguir los pasos anteriormente descritos.

4. Descripción de los archivos y las clases

4.1. ArduinoPins

4.2. Board

Este archivo solo contiene un archivo de cabecera con la declaración de la versión de la board, es decir la pc, para el farmbot existen 4 versiones, su versión mas antigua es la ramps, la segunda y tercera versión es la que se encuentra en el famrbot uniandes actualmente. y la ultima versión que incluye Stepper drivers que se comunican por SPI y que estará en el farmbot para 2020-2. Esta interfaz fue programada usando la versión 1.4 del farmbot.

4.3. Command

Esta clase es la encargada de tomar un String que es el mensaje que se envía por serial y devolver un objeto de tipo comando con el comando que se envió, ademas también tiene métodos que parten ese mismo String para obtener los parámetros que acompañan, como ejemplo si envía el comando "G00 X100 Y100 Z100", al crear un objeto de tipo Çommandçon este string, este tendrá las siguientes propiedades:

- CommandCodeEnum G00
- long X= 100
- Long Y=100
- Long Z=100

Cabe aclarar que esta clase no es la encargada de ejecutar el comando que se envió solo guarda su información.

4.4. Config

Este archivo solo contiene un archivo de cabecera con los parametros del farmbot default asi como las constantes que tienen diferentes mensajes de respuesta del farmbot (R00,R99, etc). Estos se encuentran definidos para cada una de las versiones.

4.5. CurrentState

Esta clase contiene el estado actual del farmbot es decir:

- La posición actual en los 3 ejes
- La posición de la cola de ejecución de comandos
- El estado de emergencia (si se encuentra en emergencia o no)
- La conversión de pasos a mm

Ademas esta clase permite acceder a los valores anteriormente mencionados mediante métodos get y cambiarlos mediante métodos set.Como funcionalidad adicional tiene el método "Print-QAndNewline"que imprime por serial los caracteres que se ponen al final de cada mensaje.

4.6. F09-F84 Handler

Estos son varios archivos, cada uno ejecuta la función que lleva por nombre con los parámetros que necesita para ejecutarla a los cuales accede a través del comando que recibe como parámetro en su instancia-miento y los métodos get de la clase current state. Cada una de estas funciones es de tipo "GCodeHnadler"

4.7. G00 and G28 Handler

Estos son varios archivos, cada uno ejecuta la función de movimiento que lleva por nombre con los parámetros que necesita para ejecutarla a los cuales accede a través del comando que recibe como parámetro en su instancia-miento y los métodos get de la clase current state. Cada una de estas funciones es de tipo "GCodeHnadler"

4.8. GCodeHandler

Esta clase esta escrita como un "plantilla"para las clases F y las clases G, unicamente cuenta con el comando .^{ex}ecutez establece el modelo para las demas, ya que las clases descritas en la sección **??** y 4.6 son de este tipo deben seguir su mismo modelo

4.9. GCodeProcessor

Esta clase es la encargada de recibir el comando por string, convertir ese string en un comando, tomar el comando y según su ÇommandCodeEnumçrear un objeto de tipo GCodeHandler que tiene la función especifica y correr el método execute de esta función.

4.10. MainInterface

Clase con la interfaz de usuario, esta clase puede decirse que es la principal, ya que esta crea una instancia del farmbot, ademas que es a la que se entra desde el main. Una particularidad de esta clase es que esta escrita en C++/CLI usando .NET 4.7.2 por lo que se debe tener en cuenta el tipo de clase que es y particularidades, en el siguiente link encontrara las diferencias entre C++ y C++/CLI, como se conecta con .NET y como manejar ambos lenguajes.

4.11. MovementAxis

Clase encargada del movimiento de un eje, cuenta con los métodos para saber si el eje esta en home o si ya llego a su destino. Ademas en este se incluyo la simulación de la velocidad del farmbot usando el tiempo del sistema y las constantes de velocidad par saber cuantos pasos debía moverse en una ventana pequeña de tiempo de 10 ms

4.12. Movement

Movimiento general del robot, incluye 3 variables de tipo MovememementAxis, una para cada eje. esta se encarga de que el Farmbot llegue a su destino en los 3 puntos usando los métodos de validación de MovementAxis

4.13. ParameterList

Clase con la lectura y escritura de parámetros del Farmbot. Los parametros son originalmente guardados en la EEPROM de arduino, al no tener ese recurso disponible se guardó en el disco duro del sistema usando archivos CSV.

4.14. PinClass

Clase que sustituye los pines físicos del arduino, esta clase crea objetos de tipo pin que tienen un valor y se define si son de entrada o salida etc como si estos fueran los pines físicos del arduino, pudiendo acceder a sus valores desde la clase ArduinoPins explicada en 4.1

4.15. PinControl

Clase encargada del control de pines físicos del arduino, no utilizada en este codigo, pero se mantuvo para evitar errores en declaraciones ya que el archivo cuenta con algunas constantes

4.16. Pins

Este archivo solo contiene un archivo de cabecera con la declaración de los pines del arduino en función de la versión de la board, es decir, el nombre y el numero de pin donde irían conectados los actuadores y sensores del farmbot.

4.17. StatusList

Esta clase se encargaba de escribir y leer los pines del arduino así como del pinguard usando la clase pincontrol, como pincontrol no fue implementada, esta clase tampoco. La sustitución de esta clases en modo simulación fue implementada en la 4.1

5. Probando el código

En el siguiente video puede encontrar una prueba de funcionamiento del Farmbot usando la interfaz de python, en caso de que no quiera usar la interfaz puede usar cualquier monitor serial como el de arduino y enviar los comandos manualmente teniendo en cuenta los puertos virtuales que creo en la sección 3.4, la descripción de los comandos la encuentra en la pagina del Farmbot en el siguiente link

6. Trabajos futuros

Algunos trabajos futuros que podrían ser implementados para mejorar el performance de esta interfaz son:

- **Incluir mutex:** esta clase es usada en el threading para evitar que dos hilos accedan a la misma variable al mismo tiempo.Para este caso las variables a las que podría acceder el programa al mismo tiempo desde diferentes hilos son las posiciones. Sin embargo debido al desarrollo y timing de los hilos es poco probable que estos accedan a la misma variable pero es un aspecto a tener en cuenta.
- **Validación de final de eje:**El farmbot original gracias a los encoder puede parar cuando llego al final del eje, al no tener estos en el farmbot simulator debe implementarse alguna estrategia para verificar que se llego al final,esta podría ser calcular el largo del eje en pasos y comparar con el numero de pasos dados por cada eje.