Programación Orientada a Objetos

Trabajo Práctico Integrador: C++ Avanzado

Profesor: Esp. Ing. César Aranda

Ingeniería en Mecatrónica

Universidad Nacional de Cuyo

2018

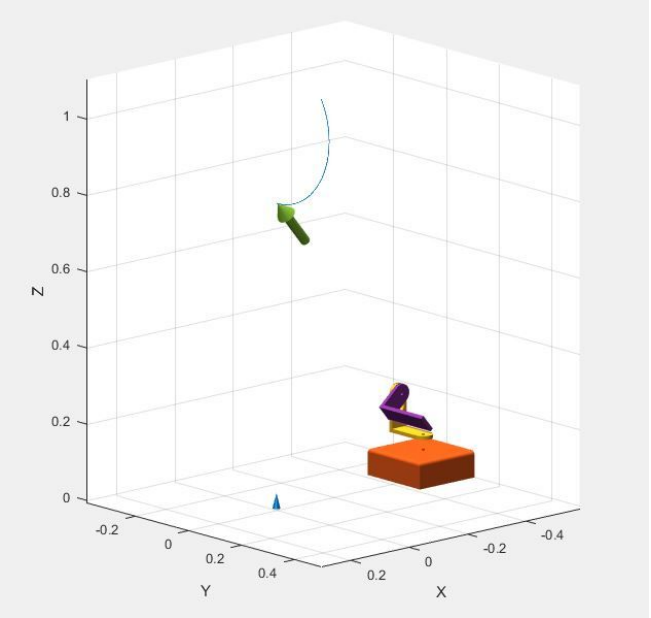
Quiroga, Martin Gabriel

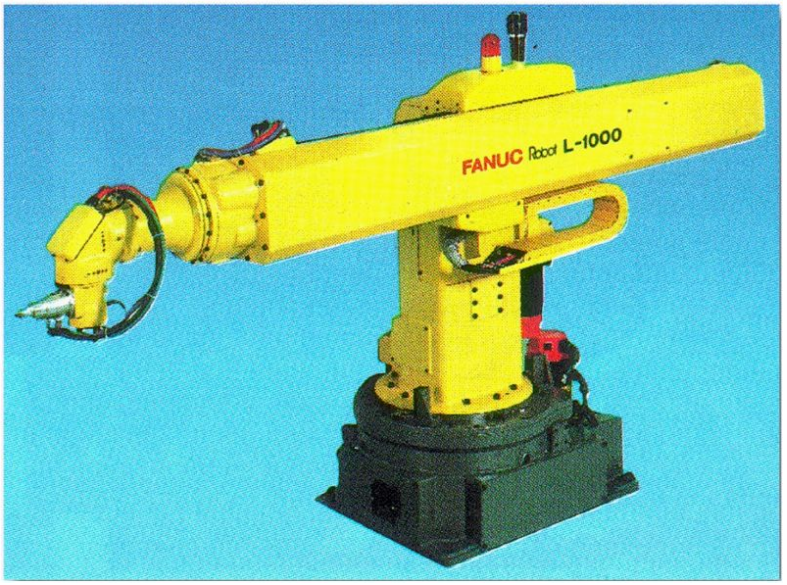
Pérez, Jerónimo

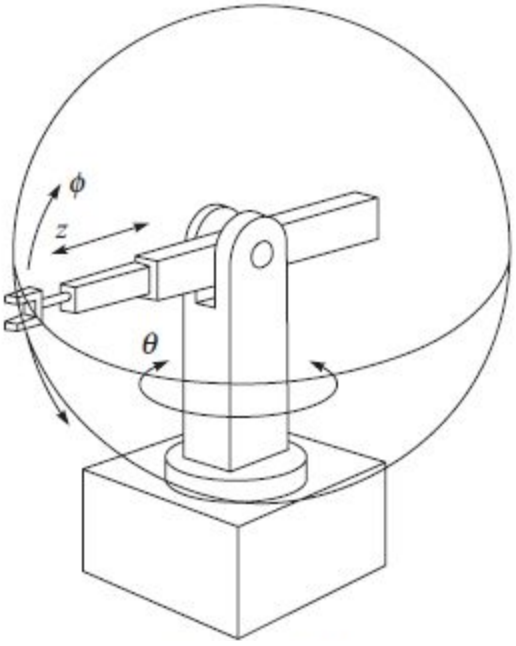
Introducción

El objetivo general del trabajo es diseñar el control de un robot con 3 grados de libertad con un efector final utilizando conocimientos de C++ adquiridos previamente y en la clase, junto con el framework Qt5.

En este caso, el robot asignado es del tipo RRP, es decir de configuración esférica, en el cual las primeras dos articulaciones son de tipo rotacional, en tanto que la tercera es de tipo prismática y virtual ya que es el punto focal de la cámara y no hay un eslabón que se desplace. El término de configuración esférica se debe al hecho de que son justamente las coordenadas esféricas, o polares, las que mejor definen la posición del efector terminal de este tipo de robots, con respecto a un sistema de referencia. Se usan en el manejo de máquinas-herramientas, soldaduras por puntos, vaciado de metales, fresado, soldadura a gas, y soldadura al arco. Particularmente, este robot tendrá como efector final una cámara fotográfica, por lo que su aplicación será prácticamente tomar fotos y grabar.







Ejemplo de implementación:

La empresa Motorized Precision ha desarrollado un brazo robótico para filmacion (con una configuración distinta a nuestro robot) y un software de control llamado “MP Studio”.

Temas investigados:

Para su elaboración se debió investigar sobre algunos temas de C++ y profundizar sobre otros ya conocidos. Los temas investigados se detallan en una lista a continuación:

- Vectores
- Stringstream
- Manejo de archivos (.txt)
- Herencia
- Modularidad
- Clases y Objetos

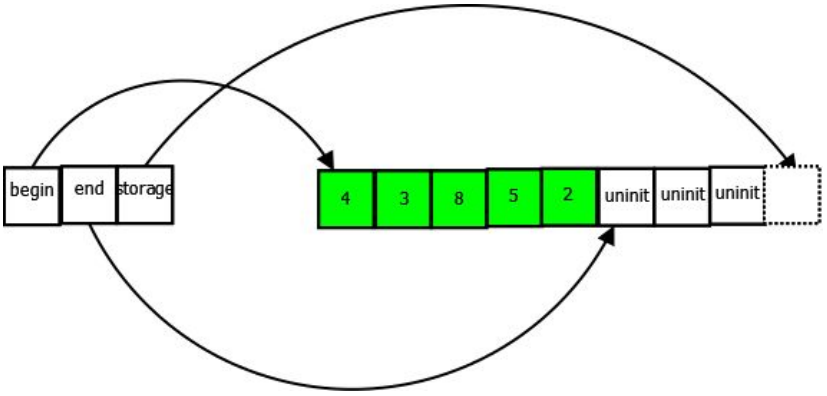
Marco Teórico

- Vectores

Los vectores son contenedores de secuencia que representan matrices que pueden cambiar de tamaño. Justo como los arreglos, los vectores usan ubicaciones de almacenamiento contiguas para sus elementos, lo que significa que se puede acceder a cada elemento usando offsets en punteros regulares a sus elementos, y de manera tan eficiente como en los arreglos. Pero a diferencia de éstos, su tamaño puede cambiar de forma dinámica, con su almacenamiento siendo manejado automáticamente por el contenedor.

En resumen, en comparación con los arreglos, los vectores consumen mayor memoria a cambio de la habilidad de administrar el almacenamiento y crecer dinámicamente de una manera eficiente.

Para poder utilizar vectores, se debe incluir la librería <vector>.



En la aplicación dada, se usaron vectores para almacenar en la base del robot los comandos que venían escritos en un archivo txt o que se introducían por la interfaz, de tal manera que cada línea del archivo de texto pasaba a ser un elemento del vector. También se utilizaron vectores para almacenar las velocidades de las articulaciones y sus respectivos tiempos luego de cada iteración, para así poder graficar la velocidad en función del tiempo.

- Stringstream

Es una clase para operar sobre cadenas. Los objetos de esta clase usan un buffer de cadenas que contiene una secuencia de caracteres. Se puede acceder a esta secuencia de caracteres directamente como un objeto cadena, utilizando la función miembro str.

Los caracteres pueden insertarse y/0 extraerse de la secuencia utilizando cualquier operación permitida tanto en input como para output.

Se debe incluir la librería <sstream> para poder aplicarlo.

Básicamente se usó stringstream para hacer los informes de las acciones que realizaba el robot, de manera que se pudo armar con strings dados en el código mismo, junto con los datos almacenados en las variables de los objetos, las cuales podían ir cambiando de un comando al siguiente.

- Manejo de archivos

  C++ proporciona las siguientes clases para realizar la salida y entrada de caracteres a/desde archivos.
  - ofstream: clase stream para escribir en archivos
  - ifstream: clase stream para leer archivos
  - fstream: clase stream para leer y escribir desde/a archivos

  Se utilizó la última clase en esta aplicación para leer archivos y almacenarlos en la
  base o para guardar en otro archivo las acciones que el robot iba realizando. Se tuvo que incluir la librería <fstream> para poder usar las funciones de esta clase, como abrir y cerrar archivos, borrarlos, escribirlos, leerlos, etc.

- **Librerías:**

  Aunque inicialmente se experimentó con librerías para la representación en 3D del robot, en la aplicación final solo utilizamos "CustomPlot" para realizar las gráficas de velocidad-tiempo.

- **QT (framework):**

  Al comenzar el desarrollo del proyecto encontramos dos caminos posibles, utilizar QT Quick que permite una interfaz gráfica más pulida pero requiere alejarse un poco de la programacion en C++ ya que esta basado en QML ó utilizar QT widgets que lleva más tiempo en uso y hay más información disponible, por lo que optamos por esta última opción

Desarrollo

Diagrama de clases: En el mismo podemos observar la relaciones entre clases, teniendo una idea general de la implementación sin necesidad de acceder al código. En este caso se incluyen sólo los atributos y métodos más importantes de cada clase ya que de otra forma sería muy difícil visualizarlo.

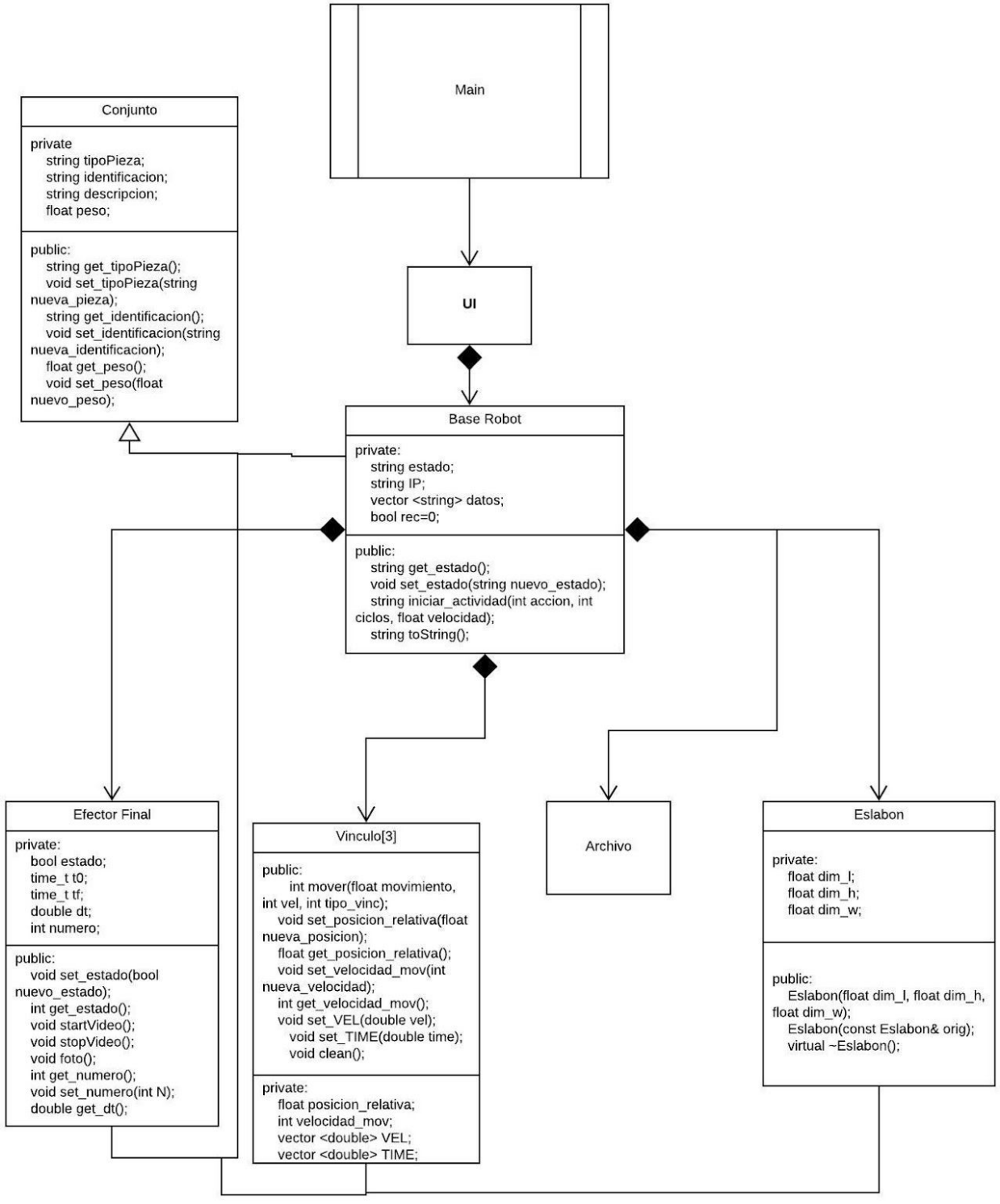


Diagrama de Secuencias: Mientras que el diagrama de clases es estático y no muestra la evolución del programa en el tiempo, el diagrama de secuencias nos permite ver la comunicación entre objetos dada una consigna específica, en este caso mostramos el diagrama para una consigna de movimiento activando el robot previamente.

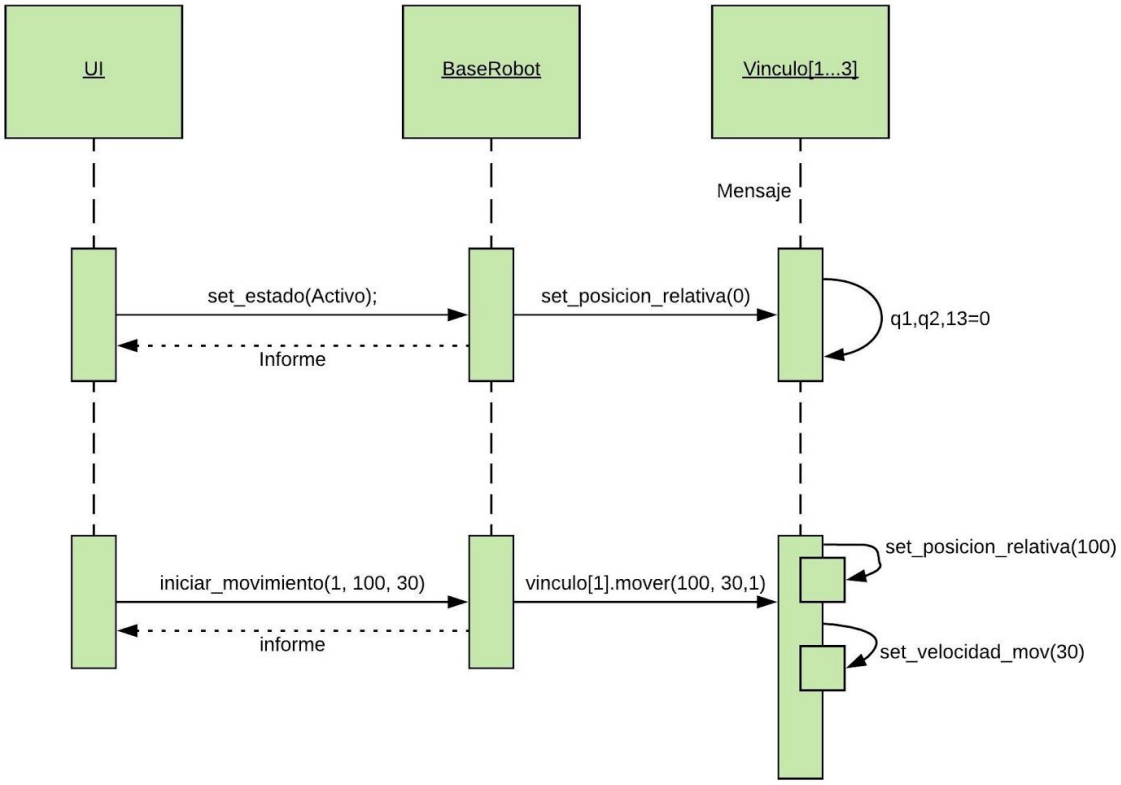
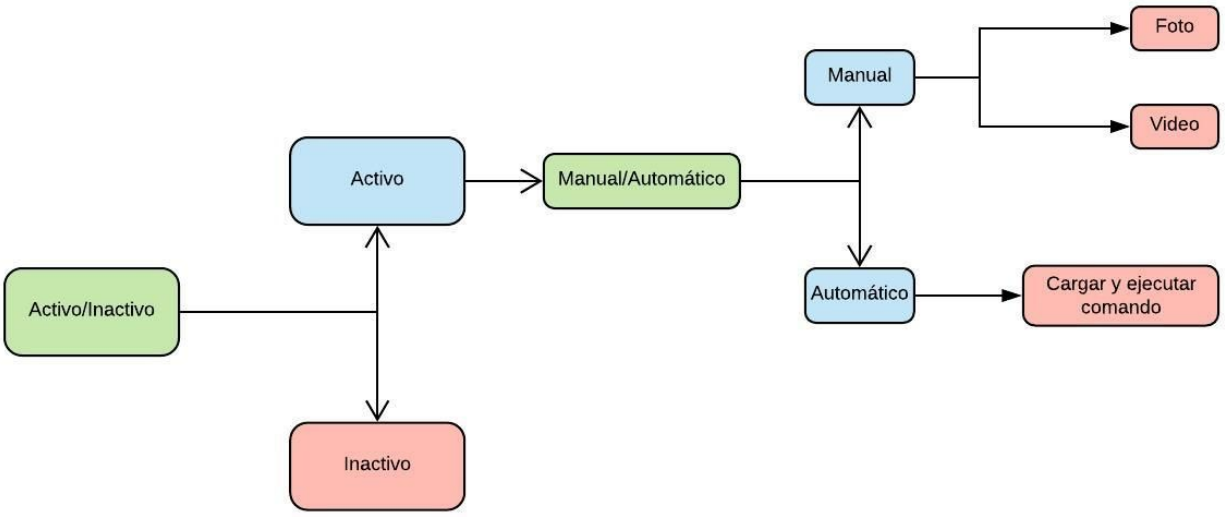
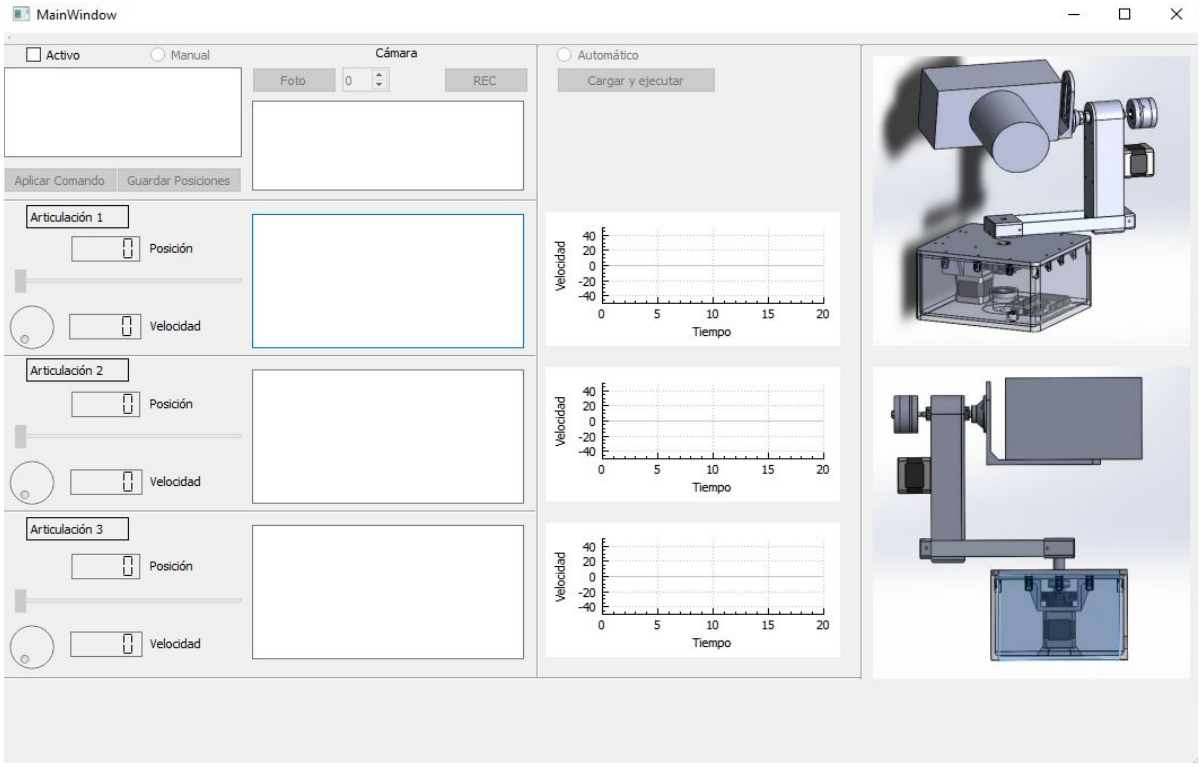


Diagrama de Actividad: El diagrama de actividades es un diagrama de flujo que nos permite seguir el proceso a realizar para llegar a una determinada acción, en este caso representamos los pasos para actuar sobre el efector final.

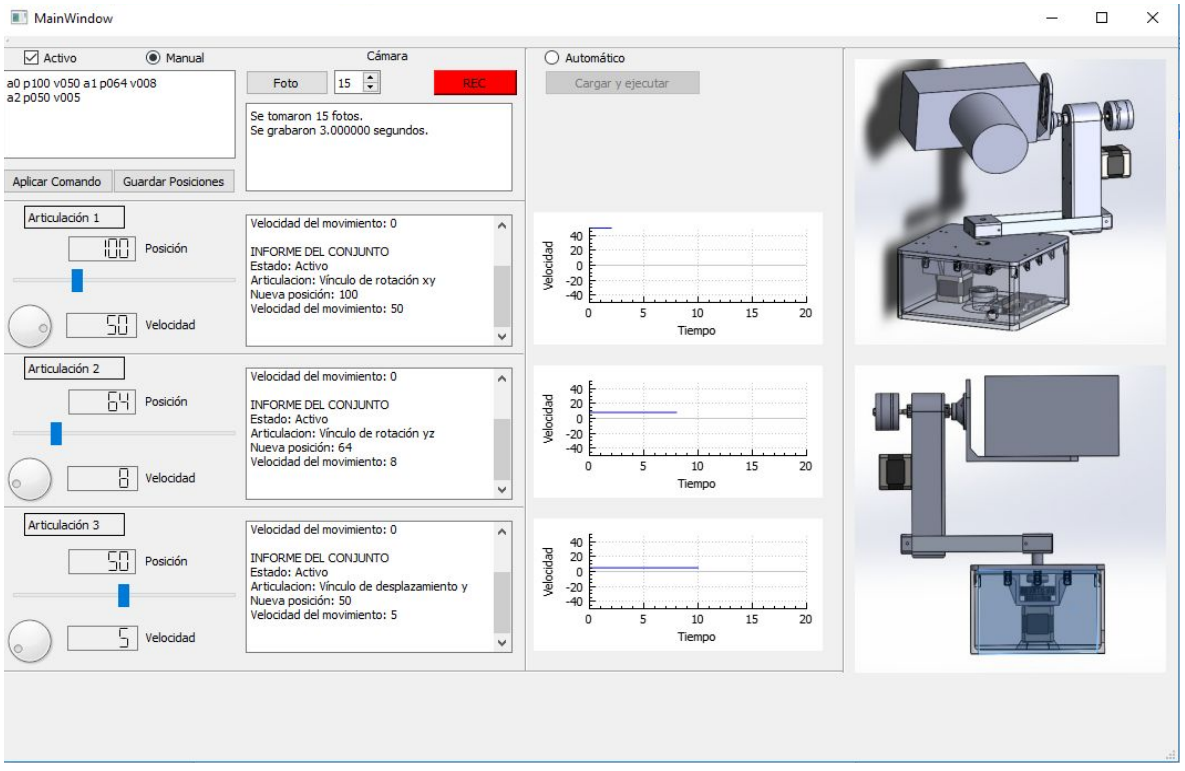


Screenshots:

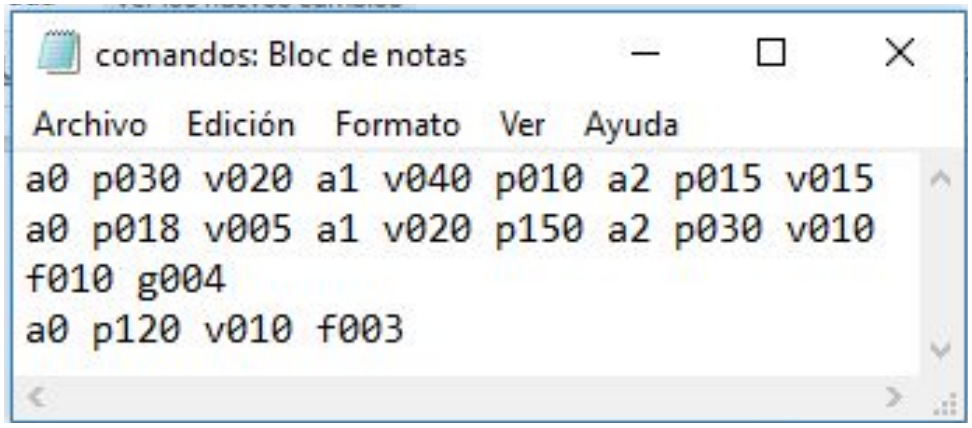
- Interfaz inicial

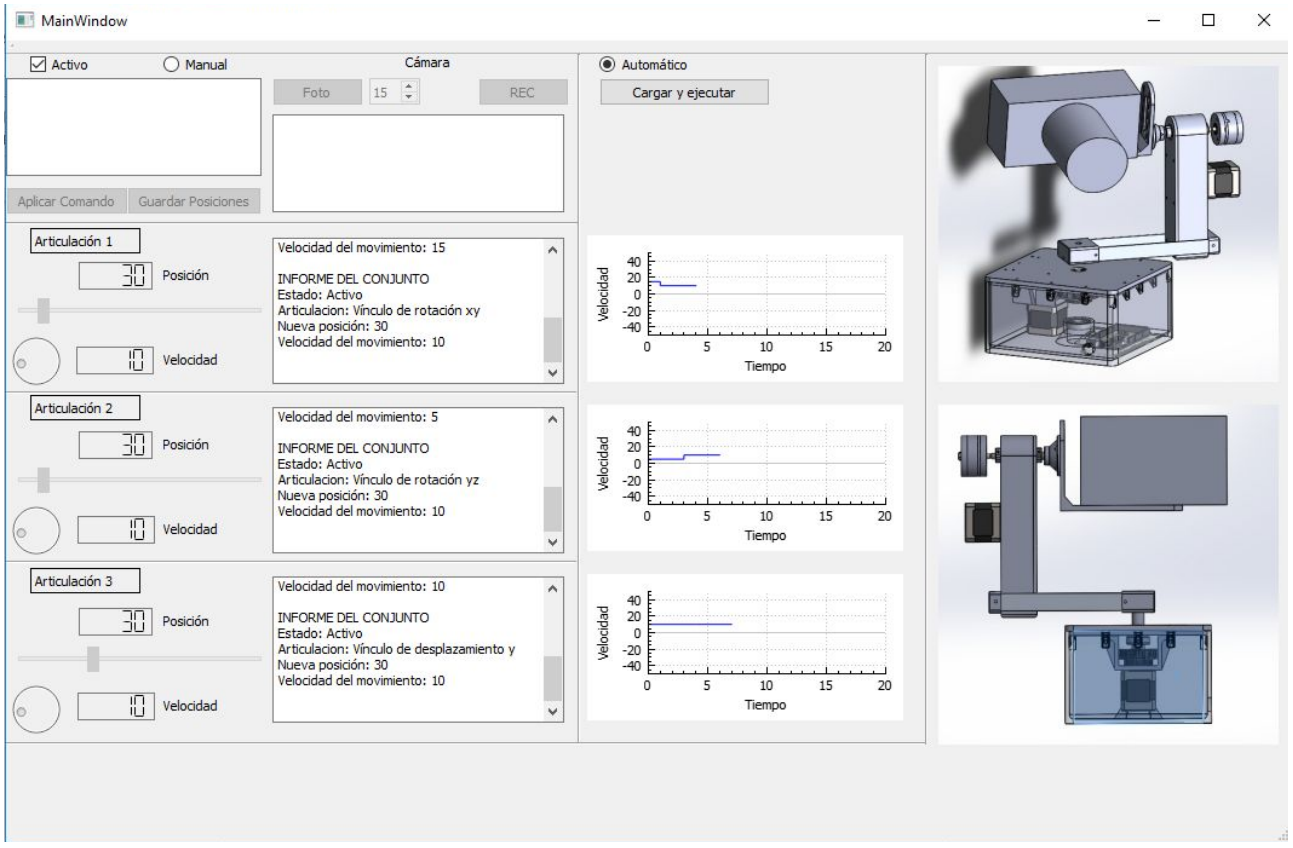


- Uso del modo manual



- Uso del modo automático





Comentarios y Conclusiones

Analizando el proceso seguido para llegar a una aplicación funcional vimos que las mayores dificultades se presentaron en aprender a utilizar la plataforma QT, ya que era totalmente nueva y está en desarrollo continuo por lo que tuvimos bastantes problemas de inestabilidad de la misma, aunque una vez solucionado esto, enlazar el código de C++ realizado en NetBeans fue relativamente sencillo y estamos muy conformes con la modularidad que logramos en la implementación.

Como posibles mejoras nos gustaría implementar un visualizador 3D para simular los movimientos del robot, pulir el diseño de la interfaz, posiblemente migrando a QT Quick y agregar funcionalidad a algunos objetos que en esta instancia están sólo como contenedores de información.

Bibliografía

www.cplusplus.com

wiki.qt.io

Qcustomplot.com

https://es.stackoverflow.com/