Hardware de Abstracción para Controlador de Motores

Presentación

Mi nombre es Jose Ramon Cruz Torres, ingeniero mecatrónico egresado de la Universidad Politécnica de Chiapas en México. exactamente de la misma generación que Eduardo y Uriel, solo que diferente carrera por supuesto.

Mi pasión es la automatización de equipos y sistemas de hardware para la solución de problemas en los que se necesita tener el control de algo. Por ende, mi herramienta principal es el software LabVIEW de National Instruments, y consiste básicamente en programación gráfica.

Actualmente me encuentro trabajando en Meta (Facebook) en el área de Reality Labs Research, dentro del equipo de Hardware Test and Automation Team. Desarrollando aplicaciones que automatizan equipos utilizados por los científicos para llevar a cabo sus múltiples investigaciones. Dentro del control de hardware, se encuentran equipos como controladores de motores lineales o rotacionales, fuentes de poder programables, cámaras, shutters, láseres, hexápodos, etc. Todos automatizados por medio de LabVIEW.

Sin embargo, el crecimiento de nuevos lenguajes de programación como Python, su facilidad de codificación y que es open source, ha hecho que el mundo científico lo utilice demasiado, por ende, los proveedores de estos equipos vienen constantemente con librerías en Python o .NET para su automatización.

La mayoría de los científicos del área de investigación utilizan Python para crear rápidamente programas que les permitan ejecutar procesos y obtener resultados de ellos para sus investigaciones. Por ende, hay una necesidad de que el equipo de automatización tenga conocimientos de dicho software.

De lo anterior nació la necesidad personal de buscar medios para aprender Python, y afortunadamente CodigoFacilito empezó a promocionar el bootcamp de python avanzado en esos meses. A pesar de no saber ni lo básico, decidí tomarlo para tener mayor responsabilidad y forzarme a aprender dicho lenguaje de programación y no postergar dicho conocimiento.

Definición del Proyecto

Durante mi estancia trabajando en Meta, me ha tocado crear varios “Engines”, es decir, módulos creados en LabVIEW para que puedan ser usados como un standalone program o puedan ser reutilizados en proyectos más grandes, donde se controlan diferentes tipos de equipos.

En algún momento creé dos engines de controladores de motores de diferentes marcas, los cuales iban a ser utilizados en una misma aplicación con más equipos automatizados y tuve que realizar una capa de abstracción para que el usuario final pudiera elegir usar uno a la vez con la misma interfaz de usuario. Por ende, sabiendo lo que he realizado en LabVIEW y observando los requerimientos del proyecto final. Decidí implementar una aplicación que ya conozco, aunque no con las mismas características como en LabVIEW, ya que se requeriría de mayor tiempo y comprensión de lo que se tiene que implementar.

Alcance del Proyecto

Ya que dicho proyecto está relacionado fuertemente con mi trabajo y será una herramienta utilizada o lista para ser utilizada a futuro, he creado diferentes alcances para que ustedes puedan observar la planificación y mejora del proyecto.

Alcance Original

Crear una aplicación donde se pueda automatizar el uso de dos marcas diferentes de controladores (PI y Newport). Y ser capaces de controlar dos motores con cada tipo de controlador. Para ello se realizará una capa de abstracción formal en Python.

Al igual se hará uso de la herramienta PyQT5 para realizar una interfaz de usuario donde se pueda tener controles e indicadores que permitan al usuario realizar acciones como: conectar, desconectar, mover a una distancia absoluta, mover a una distancia relativa positiva o negativamente, detener, seleccionar motor y seleccionar controlador. Así como mostrar al usuario mediante indicadores la posición actual de motor seleccionado, ver mediante un led virtual si el motor se está moviendo, o si el controlador está conectado o no, y obtener el status del motor en un string indicator.

Dentro del programa interno en lenguaje Python, se implementarán conceptos como:

- Módulos y paquetes
- Programación concurrente
- Funciones
- Decoradores
- OOP
- Documentación PEP8

- Pruebas unitarias
- Estructura de Datos

La programación será realizada en VS Code, herramienta principal en Meta. En Python 3.10. Y el idioma utilizado dentro del programa será totalmente en inglés. Para que pueda ser reutilizado en un futuro por el equipo de automatización, o ser tomado como base de referencia para la creación formal de futuros engines en Python.

Alcance Implementado

El código implementado se encuentra en el siguiente repositorio. Y también se puede ver un demo en el siguiente video como parte de la demostración de la implementación final del proyecto.

Cada uno de los puntos mencionados en alcance original fueron implementados exitosamente dentro del proyecto final entregado a Codigofacilito. Con excepción de los siguientes puntos mencionados, debido a los tiempos en implementación o carga de trabajo.

- El indicador de Status no fue implementado ya que en cada controlador es diferente y para evitar contratiempos se decidió ignorarlo en estos momentos.
- Pruebas unitarias no fueron realizadas para los módulos. Se requiere analizar nuevamente las clases para una futura implementación.
- No realicé la validación del formato PEP8 con ayuda de algún tool propuesto durante el curso. Solo leí las indicaciones y las consideré lo mejor que pude. Invito a mi revisor para ver si detecta errores de PEP8 mediante el uso de algún tool.

Retos

Creo que el mayor reto fue literalmente aprender un nuevo lenguaje de programación y saber utilizar el módulo PyQT5 para crear interfaces de usuario. Como mencioné anteriormente, yo soy programador en un lenguaje gráfico, y por ende es más fácil crear UI's y la programación es conectar pequeños boxes que contienen las funciones. Soy más visual, visualizo gráficamente lo que programo y veo gráficamente lo que puede pasar en cada paso. Y en parte esto de tener una idea visual de lo que hago en LabVIEW, me ayudó a comprender cada uno de los temas.

Al principio no pensaba en realizar un proyecto final. Sin embargo, al pasar las clases y ver que aún no había programado nada. Me dispuse a implementar el proyecto para poner manos a la obra todo lo que había aprendido con los maestros. Considero que fue la mejor elección que hice. Nunca me hubiese imaginado realizar un proyecto en programación escrita como el que se ha entregado aquí. Pues a pesar de recibir clases en la universidad y maestría de programación en C++, CVI y

Matlab. Nunca tuve una comprensión profunda de los lenguajes o editores para así realizar proyectos más complejos y formales utilizando programación escrita.

Alcance a futuro en Meta

A futuro y de manera más formal, se implementará una interfaz de usuario específicamente para cada controlador. A lo cual se le llamará engine, y el cual, en caso de ser reutilizado en otro proyecto, este deberá funcionar de manera adecuada y detectar que está siendo llamada desde un __main__. Por ende, no desplegará una interfaz de usuario específica del engine, sino que solo se encargará recibir los comandos provenientes de un main y se encargará de retornar los valores leídos a un Callback loop mediante queues.

Se agregarán más funcionalidades como jogging positive o jogging negative, home, enable, disable, abort, set velocity, set acceleration. Y más indicadores como el status del motor, o error de status, etc.

Area de Mejoras

A continuación se describen puntos negativos que se tienen que mejorar para un mejor desempeño del programa.

- Para la instancia del controlador Newport, no se implementaron mejoras que permitieran a la interfaz de usuario hacer stop mientras que se está ejecutando un movimiento. Esto es un problema de Threads, el cual el thread que controla el objeto de la api del controlador Newport se bloquea cuando manda un comando. Intente crear 3 instancias diferentes del objeto XPS() del controlador. Pero no funcionó.
  Pregunta para mi revisor: Tengo que crear un 3 instancias de la clase NewportHalClass()? O solo creo una y pongo en diferentes threads diferentes los métodos de la clase instanciada?
- Para la instancia del controlador Newport. El led indicador Moving no funciona correctamente debido a problemas con el Thread donde se está ejecutando. Consideraba que este problema era parte de PyQT5, pero a pesar de implementar varias opciones como crear diferentes Threads en Main para los indicadores. O simplemente utilizar QThread de PyQT5, no funcionó. Este problema también se presentó al principio con el controlador PI, y se solucionó en Qt Designer al colocar los LED's indicadores en un GroupBox cada uno.
  Pregunta para mi revisor: Ya que sucede solo con el controlador Newport. La solución estará en crear un Thread específicamente para el método dentro de la clase api de Newport? O creo una instancia nueva de NewportHalClass() y la pongo en un Thread separado?

- Hay muchas validaciones que se pueden hacer, como verificar que el usuario escriba correctamente los inputs de los ini files. O validar si algún controlador no está conectado o si la ip address está erronea.
- La interfaz de usuario se tendrá que mejorar notablemente.
- Realizar pruebas unitarias.
  Pregunta para mi revisor: Revisando mi código, específicamente donde puedo implementar pruebas unitarias? Me recomienda hacerlas en __ini__.py de los packages, o en README de Github? Porque no creo que sea posible poder hacer pruebas unitarias en el README de Github cuando lo que controlo es algo externo, y no una página web.
- **Pregunta para mi revisor:** Podría mencionar mejoras que usted visualice y así considerarlas en proyectos futuros.
- **Pregunta para mi revisor:** Podría mencionar su perspectiva del nivel de profesionalismo del proyecto realizado como programador en Python.