Integración de Sistemas. Control de un vestidor

La idea es utilizar el motor para controlar un vestidor, montado de forma que solo es accesible una sección de éste. Así se puede dividir en bloques diferenciados. Se asignan los distintos bloques a distintos usuarios. Entonces, al acceder uno de los usuarios al vestidor, el motor irá a la sección correspondiente para el usuario, pasando rápidamente por las demás y bajando la velocidad una vez llegue a la posición especificada. Una vez se complete la operación, el motor retornará a su posición original tras un tiempo de espera, que puede ser útil en caso de que en esa ventana de tiempo otro usuario quiera acceder a su sección, yendo directamente a la posición correspondiente sin tener que retornar al principio del vestidor.

Además, también se podrá controlar el motor de forma que se pueda controlar su movimiento libremente (en vez de ir a una posición predeterminada relacionada con el usuario en la base de datos). Esto puede ser de utilidad en caso de que se generase un problema con la base de datos, dando la posibilidad de controlar el motor de forma local.

REQUISITOS DEL SISTEMA PLC-SERVO

En primer lugar, hay dos maneras de controlar el motor. De forma local y a través de OPC_UA y código Python.

Existen tres POUs para facilitar esto:

→Main: Desde aquí se invoca los otros dos POUs dependiendo de la forma de control que esté seleccionada. Además, se hace una monitorización del par torsor del motor y hace saltar un stop en el variador en caso de que supere el umbral que se elija.

→Local_control: Aquí se hace el control local. Básicamente se utiliza uno de los potenciómetros para controlar la velocidad y dirección del motor, un interruptor para encender o apagar el motor y una entrada del PLC para reiniciarlo.

→Ua_control: Aquí se hace el control mediante OPC_UA y archivos de Python. Esto se controla desde una interfaz creada con Qt.

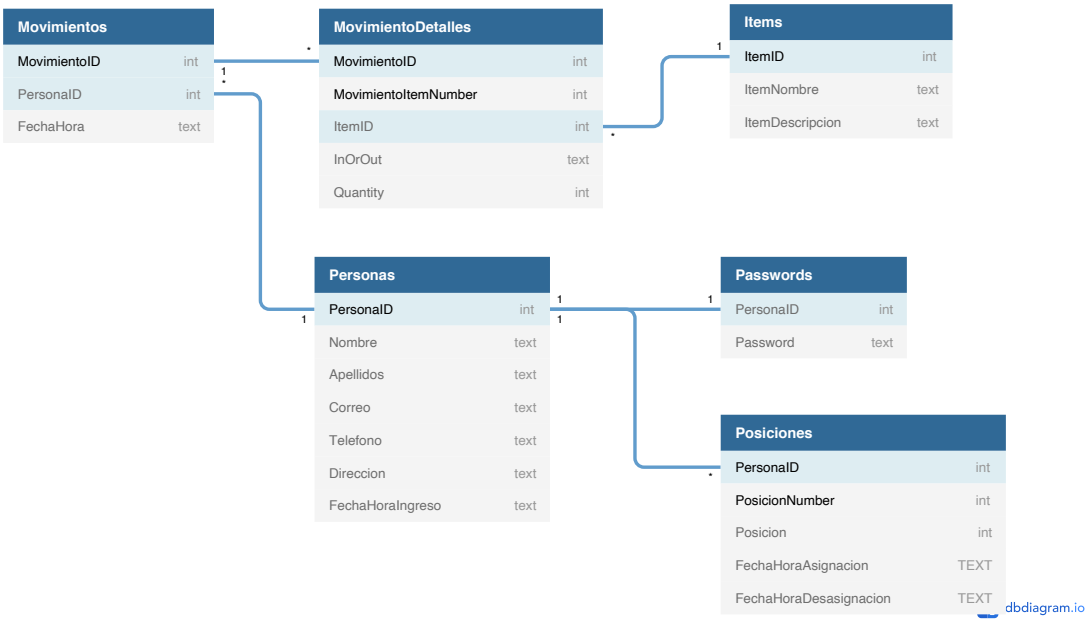
El sistema permitirá posicionamiento absoluto, que se establece en el POU Ua_control. Permite entonces movimientos a posiciones absolutas, y la posición a la que se mueva será la correspondiente de esa persona en la base de datos.

Mediante control local se permite movimiento relativo, que se implementa utilizando dos interruptores y un potenciómetro, que se asignan a PDOs. Existe una zona muerta donde el motor no se mueve y una vez en la zona de funcionamiento se regula la velocidad de forma lineal hasta llegar a la máxima, que está definida como variable global. En un principio es 200, pero se puede modificar utilizando la interfaz gráfica.

Desde las entradas del PLC se puede reiniciar el dispositivo y desactivar el bloqueo causado por un torque que supere el umbral.

REQUISITOS DEL SISTEMA DE BASE DE DATOS

La base de datos consiste en 6 tablas.



Se almacenan todos los usuarios en la tabla Personas. Las personas pueden tener 0 o 1 contraseña guardada en la tabla Passwords, que está encriptada con bcrypt desde el código de Python con la librería passlib, con la que se accede más funciones dentro de la interfaz gráfica.

Cada vez que se realiza un movimiento a una posición absoluta (control con Python) se registra en la tabla Movimientos, y si se introduce o retira uno o más ítems se deja registrado en la tabla MovimientoDetalles.

La base de datos contempla la posibilidad de hacer un cambio en la posición que una persona tiene asignada. Por eso las posiciones se guardan con la fecha en la que se asigna y en la que se desasigna. De esta forma queda registrado quién tenía acceso a qué sección en cada momento.

En el archivo de Python que interactúa con la base de datos hay varias funciones para sacar información (los ítems que hay en una posición, todos los movimientos de una persona, la posición asignada, los datos personales…) y para alterarla e introducir nuevos datos (nuevas personas, contraseñas, movimientos, nuevas posiciones, detalles de movimientos…). Además, si no existe una base de datos con el nombre indicado, se crea una nueva.

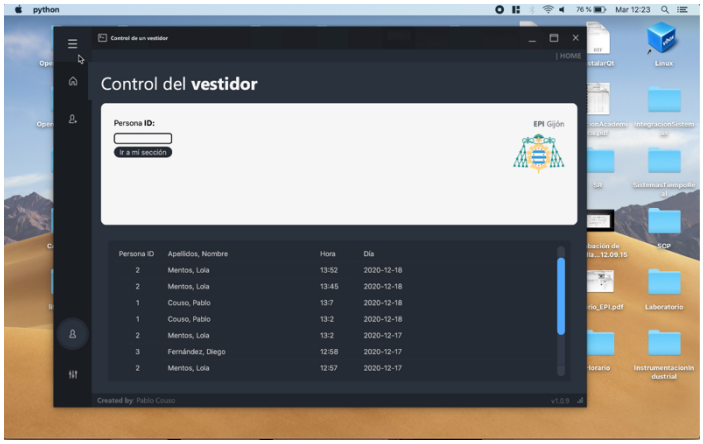
Los campos de FechaHora se guardan como texto y se tratan en Python con la librería datetime.

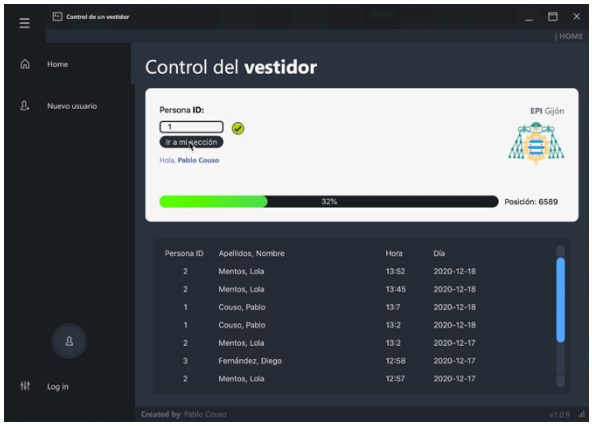
REQUISITOS DE LA INTERFAZ HUMANO-MÁQUINA

La HMI se implementa utilizando Qt mediante la librería para Python PyQt5. Con el diseñador de interfaces gráficas QtDesigner se crean archivos con formato .ui, que luego se convierten a archivos .py con el comando pyuic5 en consola para implementarlos en el programa.

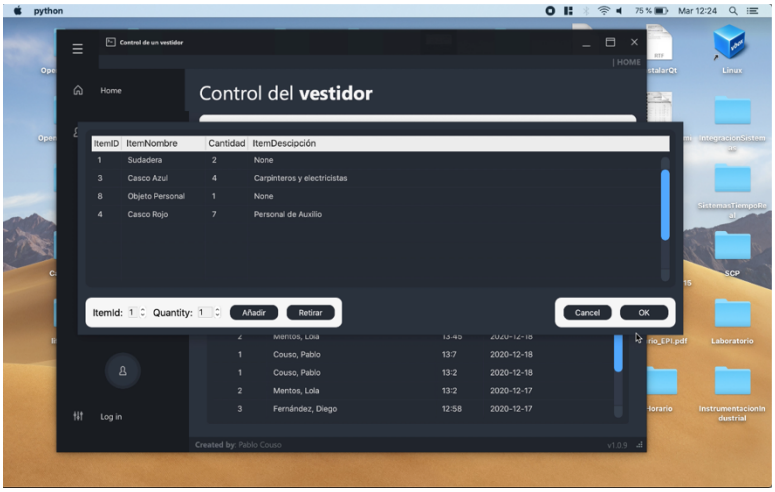
Hay una ventana principal que tiene tres pestañas disponibles.

En la primera, "Home" hay una tabla donde se muestran lo últimos 10 movimientos realizados y un cuadro de texto donde se introduce el id de una persona. Si el id es correcto muestra el nombre de la persona y activa el botón para que pueda ser pulsado. Una vez pulsado, si la persona tiene una posición asociada se manda mediante OPC la orden para que el motor vaya a la posición que esa persona tiene asociada. Mientras va a la posición se muestra el progreso con una barra de porcentaje, además de la posición actual a la derecha.



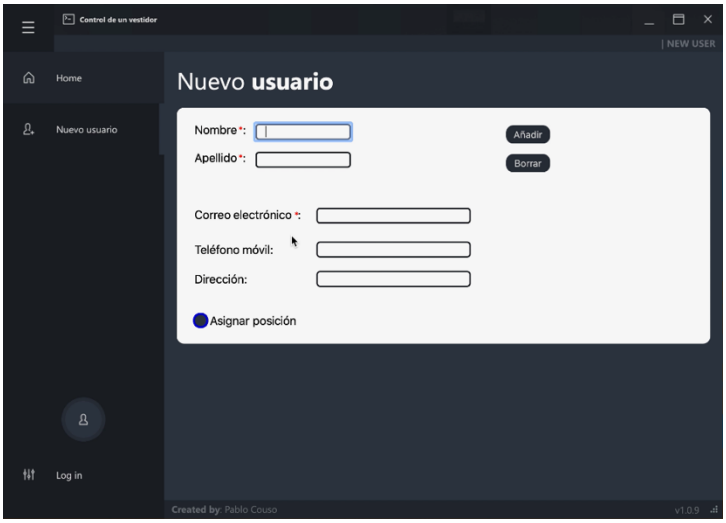


Una vez el motor llega a la posición salta otra ventana donde se muestra una tabla con todos los ítems guardados en esa posición. Aquí se pueden añadir y retirar varios ítems. Una vez se acepte, se registran los detalles del movimiento en la base de datos. En este momento, se puede introducir el id de otra persona y repetir el proceso. Si no se introduce ningún id válido en una ventana de 10 segundos (se puede configurar el tiempo en el script de Python) el motor vuelve a la posición cero.

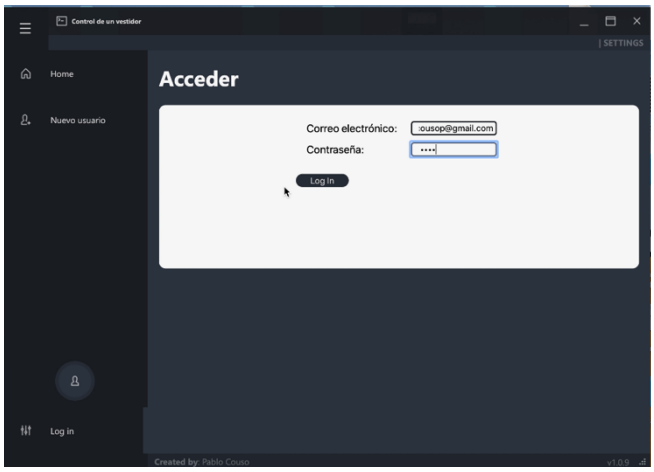


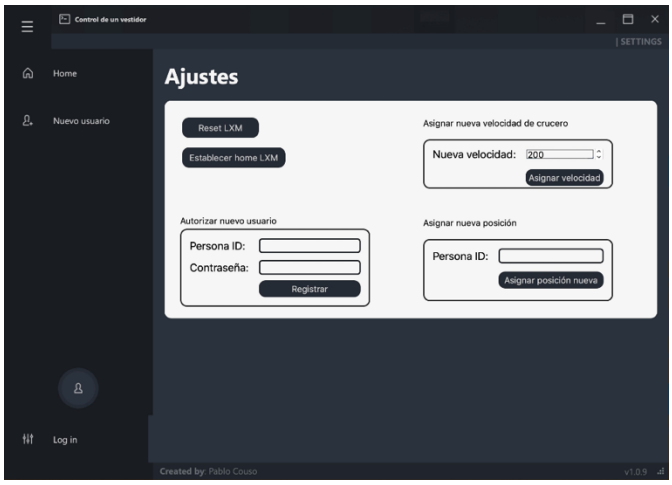
Además, en esta ventana se indicará si salta una orden de parada por un torque excesivo o si se cambia a control local, impidiendo además al usuario intentar acceder a su posición. Mientras se controla mediante la interfaz el variador, se comprueba cada 2 segundos que todo va bien, en caso de stop o control local, se cambia a cada 300 milisegundos.

Otra de las pestañas disponibles es “Nuevo usuario”. Aquí hay varios campos de texto y un radiobutton para meter información y seleccionar o no la opción de asignar una posición a esa persona que se está registrando. Una vez se rellenen al menos los campos obligatorios y se acepte, aparecerá el id para esa nueva persona y la posición (si es que se asigna una) y se registrará todo en la base de datos.



La tercera pestaña disponible es “Log in”. Aparecen dos campos de texto y un botón para acceder. Una vez rellenado se comprueba si la contraseña concuerda con la guardada en la base de datos. Si es así se pasa a una nueva pestaña. Aquí hay varias opciones: Modificar la velocidad máxima del motor (entre 100 y 500), reiniciar y mandar orden de establecer la posición cero al variador, crear una contraseña para una persona que ya esté registrada en la base de datos y asignar a una persona una nueva posición.



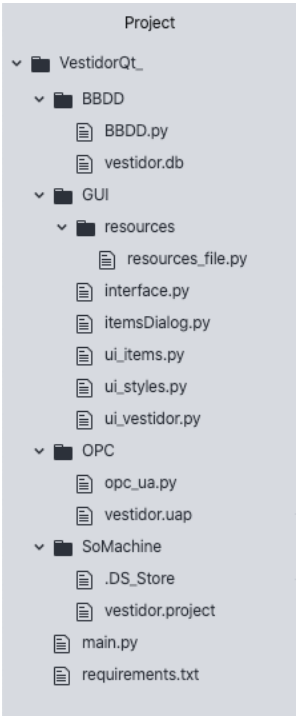


Ua Expert

Desde Ua Expert se crea un servidor que permite a la aplicación leer y escribir 10 variables del PLC. Se utiliza para el encendido/apagado, reinicio, home, comprobar control local/Python o stop, seleccionar la posición a la que el motor debe ir, monitorizar la posición actual y cambiar la velocidad máxima.

Estructura del código

Los distintos archivos se dividen en distintas carpetas. Los archivos necesarios para que funcione el programa son los que se ven en el árbol de carpetas, aunque también existen más archivos de los que se parte para conseguir los finales (archivos .ui o .qrc que se convierten a .py).



Empezando por abajo, el archivo requirements.txt contiene las librerías de Python a instalar para ejecutar el programa. Al tenerlas guardadas de esta manera, la instalación se vuelve muy simple ($pip install -r requirements.txt).

El archivo main.py es el que debe ejecutarse para inciar el programa, import las clases creadas en BBDD/BBDD.py, OPC/opc_ua.py y GUI/interface.py. Básicamente sirve como puente entre el código para el control de la base de datos y la conexión con OPC y la interfaz gráfica de la aplicación. La mayoría de sus funciones son wrappers de los dos primeros archivos que se invocan desde interface.py. De esta manera, el código está más encapsulado y por tanto un cambio en cualquiera de las partes sería más cómodo de llevar a cabo.

En la carpeta SoMachine se encuentra el archivo del proyecto para el PLC, es necesario cargarlo sobre éste para que tanto el control local como el control por OPC UA puedan funcionar correctamente.

Además del archivo opc_ua.py, en la carpeta OPC está vestidor.uap, que se abre con Ua Expert y permite la creación del servidor que nos da las funcionalidades ya mencionadas en el anterior apartado.

En GUI residen todo el código que tiene que ver con la interfaz gráfica de la aplicación, que se maneja desde interface.py. Aquí se importan todos los demás archivos de la carpeta y se implementa una clase que toma la propia clase creada en main.py como argumento, mediante la cual se podrá acceder a los métodos que se habían definido para interactuar con la base de datos y OPC, así como algunos atributos que sean de utilidad. Los scripts resources_file.py, ui_items.py y ui_vestidor.py salen directamente de convertir otros archivos provenientes de QtDesigner, aquí se definen los recursos usados (como iconos o fuentes de texto), la interfaz gráfica de la ventana que emerge mostrando los ítems de una posición concreta y la interfaz gráfica de la ventana principal, respectivamente.

Por último, en BBDD está lo relacionado con la base de datos. En el código de Python se define una clase que contiene cantidad de métodos para interactuar con la base de datos que también está en esta carpeta, vestidor.db . Desde aquí se tratan los campos FechaHora, que se reciben de la base de datos como text y se convierten a datetime. Además, en este script es donde se encriptan y más tarde se comprueban las contraseñas asignadas a determinadas personas.

En la entrega de archivos, se incluirán además de los que se mencionan aquí algunos que fuesen útiles para llegar a la versión actual de la aplicación (archivos relacionados con la interfaz gráfica).