Laboratorio SECO

Álvaro Gutiérrez

14 de febrero de 2024

Índice

1. Introducción

Actualmente, el laboratorio de la asignatura **Sistemas Electrónicos de Control** dispone de 5 puestos de Telelaboratorio (ver Figura 1.1), a los cuales los alumnos de la asignatura pueden acceder remotamente para la realización de las prácticas de la asignatura.

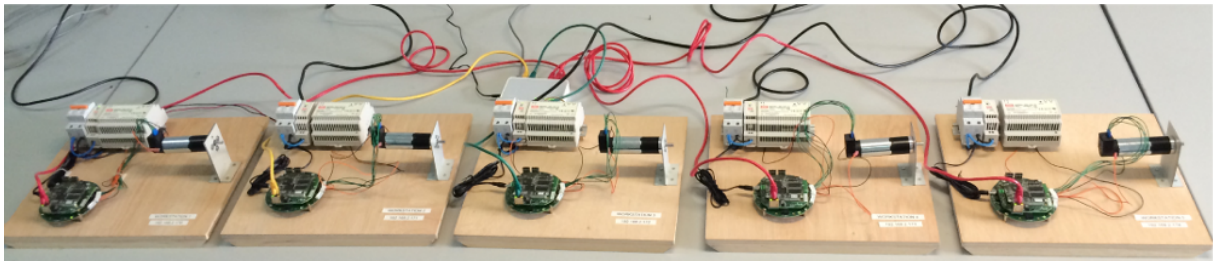

Figura 1.1: Los cinco puestos del laboratorio.

La aplicación software *secoStudentsApp* se conecta con los puestos del laboratorio a través de un servidor de colas. Este servidor se encarga de encolar los trabajos enviados por los alumnos y ejecutarlos de manera secuencial.

En la Sección 2 de este documento se explica el hardware existente en el laboratorio. En la Sección 3 se describe la instalación y puesta en funcionamiento del software para comunicarse remotamente con los puestos del laboratorio.

2. Hardware

Cada puesto de laboratorio (ver Figura 2.1) consta de los siguientes elementos:

- Protección de alimentación.
- Fuente de alimentación 5V.
- Fuente de alimentación 12V.
- Conjunto Motor, Encoder, Reductora:
  - Motor DC con escobillas A-max 32 12 V. [1]
  - Reductora Planetaria GP 32A 23:1. [2]
  - Encoder HEDS 5540 de 500 pulsos por vuelta. [3]
- Hardware de control de motores (ver Figura 2.2a).
- Cámara web

El hardware de control de motores consta de 3 tarjetas diseñadas específicamente para este laboratorio:

- Una tarjeta con un microcontrolador ARM y capacidad de controlar cuatro motores (ver Figura 2.2b).
- Una tarjeta con un procesador ATMEL con linux (ver Figura 2.2c).
- Una tarjeta que se encarga de extraer la periferia (ethernet, USB, ...) de la tarjeta anterior (ver Figura 2.2d).

El funcionamiento del hardware de control será transparente a los alumnos durante el desarrollo del curso. Sin embargo, tal y como se verá en la Sección 3 es importante tener en cuenta que de los cuatro motores que puede llegar a controlar el hardware de control, únicamente el Motor 3 se encuentra conectado para el desarrollo de la asignatura.

[1] http://www.robolabo.etsit.upm.es/asignaturas/seco/apuntes/motorDC.pdf
[2] http://www.robolabo.etsit.upm.es/asignaturas/seco/apuntes/reductora.pdf
[3] http://www.robolabo.etsit.upm.es/asignaturas/seco/apuntes/encoder.pdf

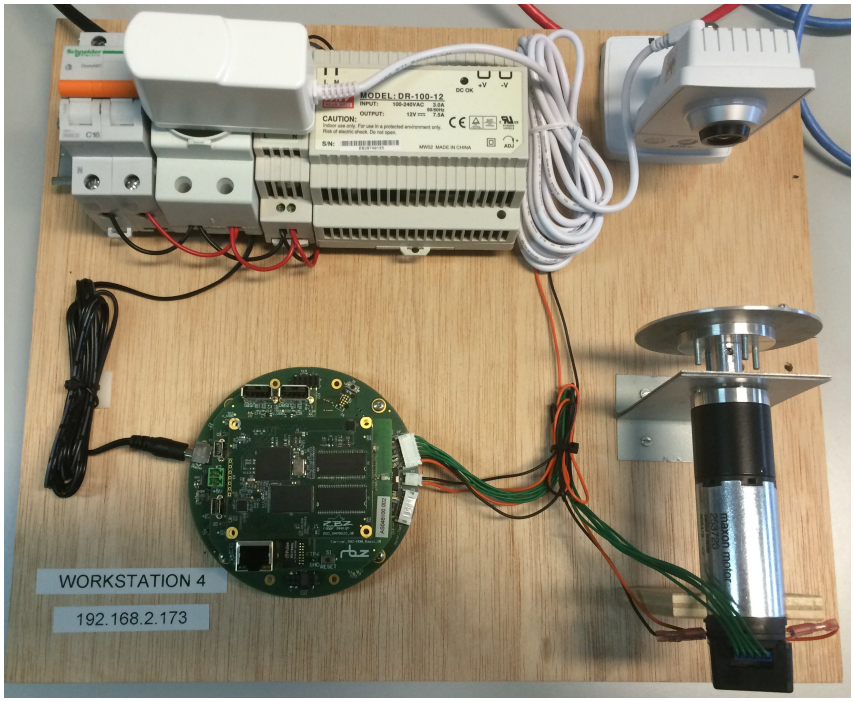

Figura 2.1: Puesto de laboratorio.

3. Software

secoStudentsApp es una aplicación que permite controlar de forma remota los motores existentes en el laboratorio de la asignatura **Sistemas Electrónicos de Control**.

3.1. General

secoStudentsApp corre en Sistemas Operativos Linux, aunque puede ser ejecutado en Sistemas Operativos Windows y Mac OS X al estar desarrollado en QT. Dichas modificaciones no se encuentran en este manual y no son materia de la asignatura, por lo que en el caso de que el alumno decida aventurarse a trabajar con dichos sistemas operativos, será él mismo quien deba resolver los problemas que surjan por su cuenta.

3.2. Librerías

Dependiendo de la distribución que tengamos, las versiones de las dependencias varían. En esta Sección presentamos las dependencias necesarias para la ejecución de *secoStudentsApp* mientras que en la Sección 3.3 se plantean unos ejemplos de su instalación.

Las dependencias necesarias para la instalación son:

- make $\geq$ 3.8
- g++ $\geq$ 4.3
- qmake $\geq$ 4.6
- qt5 $\geq$ 5

Las versiones mínimas mostradas anteriormente han sido comprobadas para el correcto funcionamiento de *secoStudentsApp*, por lo que no existe garantía de funcionamiento para versiones anteriores. Dependiendo de la distribución, existen diferentes formas de obtener cada una de las versiones de las librerías disponibles en los repositorios. Para distribuciones Debian y derivados (Ubuntu, por ejemplo), podemos ver las dependencias que tenemos disponibles mediante el siguiente comando:

```
apt-cache search <NOMBRE_DE_LA_LIBRERIA>
```

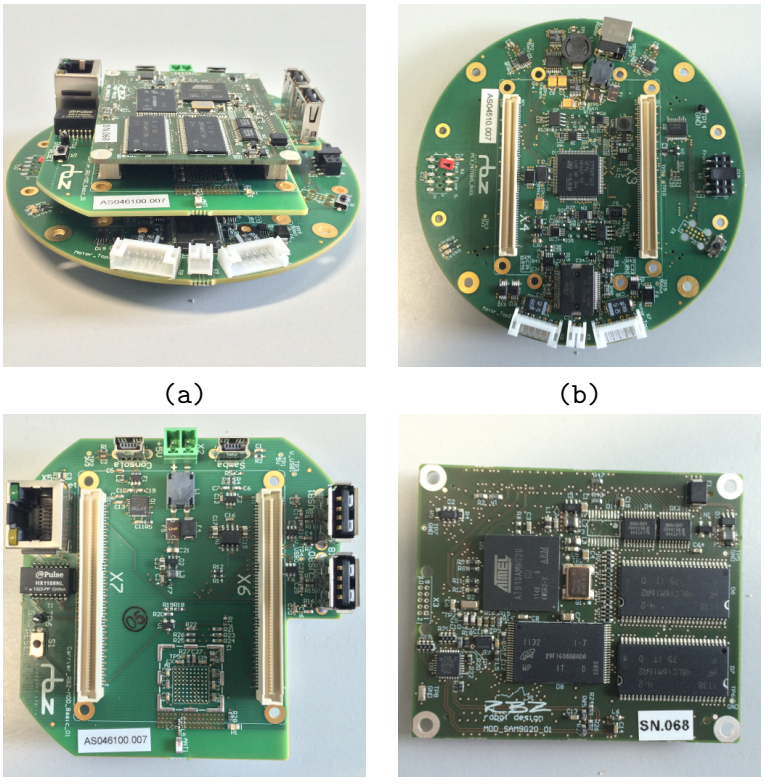

Figura 2.2: Tarjetas electrónicas del puesto del laboratorio: (a) Hardware de control de motores, (b) tarjeta de control de motores, (c) tarjeta *Carrier* de control de la periferia y (d) tarjeta con el módulo linux.

De tal manera, si por ejemplo queremos saber cuales son las versiones disponibles de g++, ejecutaremos:

```
apt-cache search g++
```

Esto nos dará una lista de librerías existentes en el repositorio. Habrá que escoger una de las librerías que cumplan los requisitos mencionados anteriormente. En la Sección 3.3 mostramos unos ejemplos.

3.3. Instalación de dependencias

Aquí vamos a presentar los requisitos necesarios para unas distribuciones Ubuntu y Debian. Para versiones diferentes de la misma distribución, o el resto de distribuciones, es necesario localizar las versiones específicas de cada paquete.

Ubuntu 18.04:
En línea de comando ejecutar:

```
sudo apt-get install make g++ libqt4-dev
```

Ubuntu 20.04:
En línea de comando ejecutar:

```
sudo apt-get install make g++ qtbase5-dev
```

Ubuntu 22.04:
En línea de comando ejecutar:

```
sudo apt-get install make g++ qtbase5-dev
```

Debian:
En línea de comando, como superusuario ejecutar:

```
sudo apt-get install make g++ qtbase5-dev
```

En el caso en el que nuestra versión de la distribución no posea alguna de las versiones de las librerías especificadas, el comando `apt-get` mostrará un error por pantalla. En este caso será necesario localizar cual es la librería errónea y substituirla por la versión correcta. Para ello hágase uso del comando `apt-cache search` tal y como se ha descrito en la Sección 3.2.

3.4. Descarga y compilación

Una vez instaladas todas las librerías es necesario descargarse el software de http://www.robolabo.etsit.upm.es/sul
Para compilar el simulador es necesario ejecutar la primera vez las siguientes instrucciones:

- Descomprimir el paquete:

```
tar -xvzf secoStudentsApp\_<version>.tgz
```

- Ir al directorio donde se ha descomprimido:

```
cd secoStudentsQueueAppExported
```

- Ejecutar:

```
qmake
```

  En Ubuntu 18.04 ejecutar:

```
qmake-qt4
```

- Ejecutar:

```
make
```

3.5. Prueba de compilación

Para comprobar que *secoStudentsApp* se ha compilado correctamente ejecutar:

```
./secoStudentsApp
```

Deberán aparecer 3 pantallas: una con un eje de coordenadas, otra con una ventana que muestra imágenes de la cámara web y otra con una interfaz de configuración con 5 pestañas: *General*, *Motor*, *Controller*, *Perturbation* y *Communication* (ver Figura 3.1).

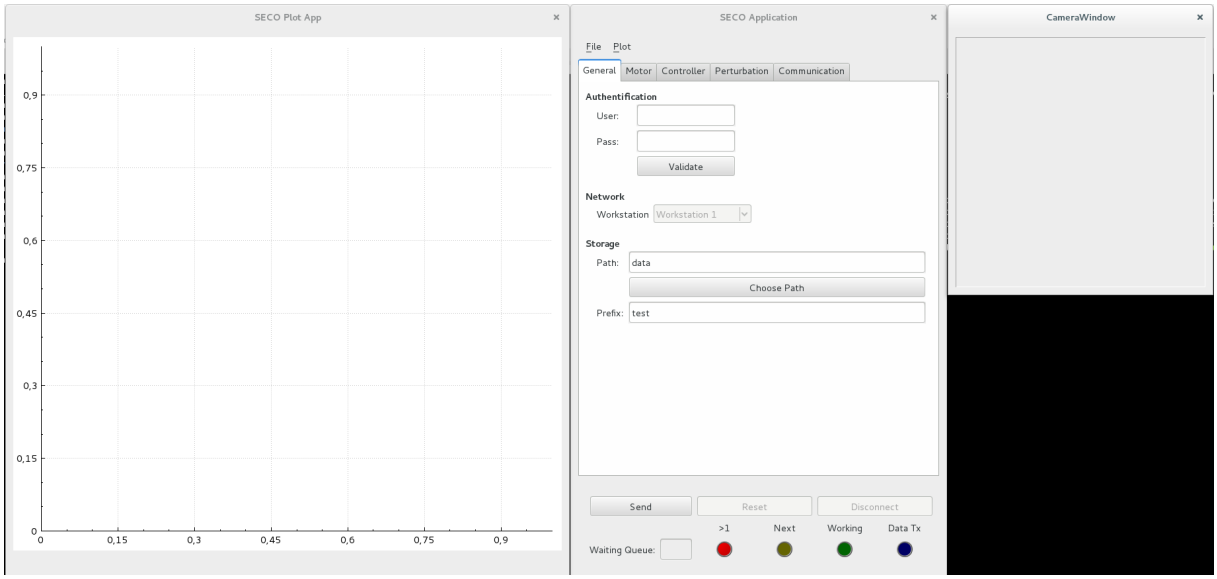

Figura 3.1: Pantalla de inicio de la aplicación *secoStudentsApp*.

Si esto es así, la compilación habrá sido correcta.

3.6. Descripción de secoStudentsApp

Como se ha comentado previamente, la interfaz de configuración consta de cinco pestañas de configuración. En las siguientes secciones se detalla cada una de ellas.

3.6.1. General

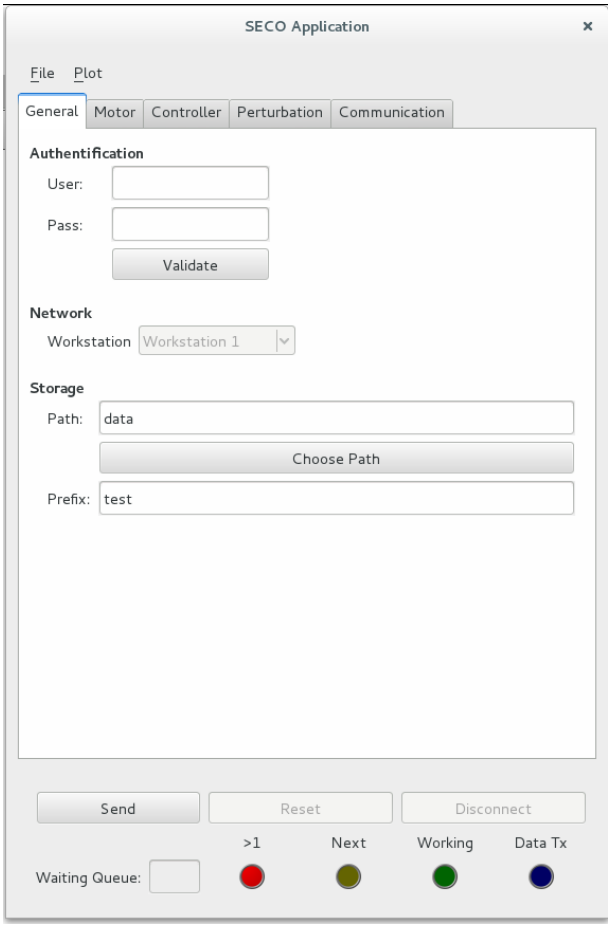

Figura 3.2: Pestaña de configuración General de la aplicación *secoStudentsApp*.

En esta pestaña se configuran los parámetros generales de *secoStudentsApp*, dividido en 3 bloques (ver Figura 3.2):

- **Authentification**: Este bloque es utilizado para autentificar al alumno con el laboratorio y comprobar que va a comenzar a trabajar en el horario establecido previamente.

  Este bloque aún no está implementado.

- **Network**: En este bloque se especifica el puesto al cual el alumno va a conectarse. Debido a la existencia del servidor de colas, la aplicación tiene bloqueado el acceso a esta pestaña, y será el servidor de colas el encargado de dar paso a cada puesto una vez que entre a ejecución el trabajo del alumno.

- **Storage**: En este bloque se especifica cual es la ruta en la que el software almacenará los ficheros de datos (`Path`, por defecto *data*), así como el prefijo de los ficheros (`Prefix`, por defecto *test*). Los ficheros serán almacenados en el directorio `Path`, con el prefijo `Prefix`, de tal manera, que todos los ficheros tendrán el siguiente formato: `<Path>/<test>-MOTOR3<VAR>`, siendo `<VAR>` alguna variable definida en la pestaña *Communication*.

  NOTA: Es importante tener en cuenta que cada vez que se realiza una ejecución, *secoStudentsApp* borra los ficheros ubicados en `Path` con prefijo `Prefix-`. Por lo tanto, conviene modificar el prefijo cada vez que se realice un experimento.

3.6.2. Motor

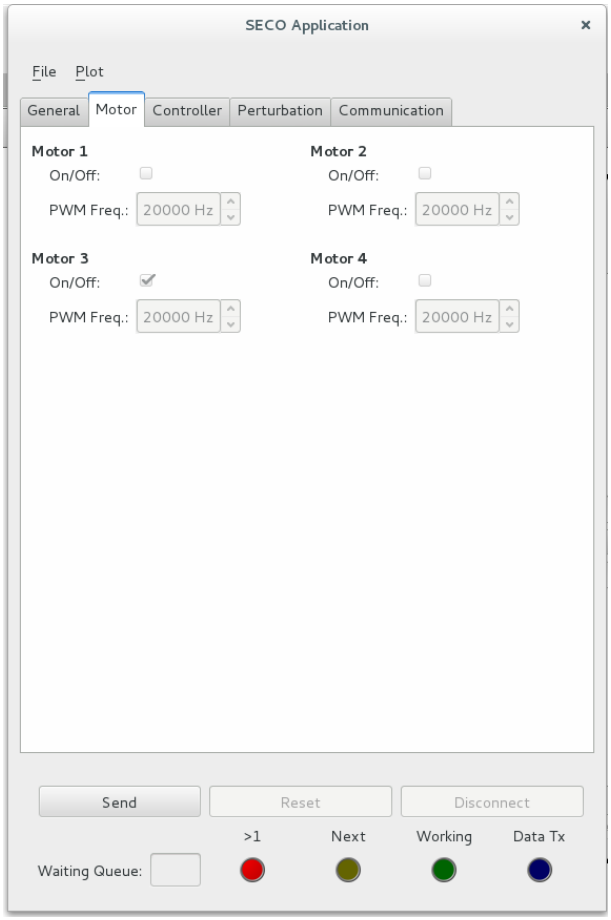

Figura 3.3: Pestaña de configuración de los Motores de la aplicación *secoStudentsApp*.

En la pestaña motor, se configuran los motores activos y la frecuencia del PWM (ver Figura 3.3). Téngase en cuenta que para la asignatura de Sistemas Electrónicos de Control, únicamente el Motor 3 se encuentra conectado y por lo tanto activo en la aplicación. Además, la configuración de la frecuencia del PWM ha sido fijada a 20 kHz.

3.6.3. Controller

En esta pestaña (ver Figura 3.4) se especifican los siguiente parámetros generales del controlador:

- **Periodo de muestreo**: La aplicación solicita el periodo de muestreo en milisegundos.
- **Tipo de controlador**: PID único y por defecto.
- **Variable de control**: Posición o Velocidad.
- **Señal de referencia**: Delta de Kronecker, Escalón, Rampa, Parábola, Seno, Coseno y Trapezoidal.
- **Valores de la señal de referencia**: Téngase en cuenta que se utilizan 4 variables (*Var1*, ..., *Var4*) para definir la señal de referencia, con diferente significado para cada una de ellas y conforme a la Tabla **??**.

Además, en esta pestaña se definen los parámetros del controlador definido. Téngase en cuenta que la arquitectura software se ha desarrollado con la intencionalidad de cubrir todos los posibles lazos de control a implementar dentro de la asignatura. Es por ello que se han definido 4 lazos de control (ver Figura 3.6): Directo (DI), Paralelo (PA), Prealimentado (FF) y Realimentado (FB). Para cada uno de los lazos, un PID se ha definido con los siguientes parámetros K_p, K_i y K_d. Adicionalmente se ha definido un parámetro de *WindUp*, aunque no es accesible para los alumnos.

Figura 3.4: Pestaña de configuración de los Controladores de la aplicación *secoStudentsApp*.

3.6.4. Perturbation

En esta pestaña (ver Figura 3.7) se especifican los parámetros de la perturbación ($w(k)$), en caso de existir, en relación al diagrama de la Figura 3.8:

- **Señal de referencia**: Delta de Kronecker, Escalón, Rampa, Parábola, Seno, Coseno y Trapezoidal.
- **Valores de la señal de referencia**: Téngase en cuenta que se utilizan 4 variables (*Var1*, ..., *Var4*) para definir la señal de referencia, con diferente significado para cada una de ellas y conforme a la Tabla **??**.

3.6.5. Communication

La pestaña de comunicación (ver Figura 3.9) es la encargada de seleccionar cuales son los datos que queremos obtener del motor y controlador para su posterior estudio.

Señal	Ecuación	Var1	Var2	Var3	Var4
Delta	$r(kT) = A\,,\, k = 0; r(kT) = 0\,,\, \forall k \neq 0$	A			
Escalón	$r(kT) = A$	A			
Rampa	$r(kT) = A \cdot kT$	A			
Parábola	$r(kT) = 1/2 \cdot A \cdot (kT)^2$	A			
Seno	$r(kT) = A \cdot sin(\omega \cdot kT)$	A	ω		
Coseno	$r(kT) = A \cdot cos(\omega \cdot kT)$	A	ω		
Trapezoidal	Ver Figura 3.5	A	t_1	t_2	t_3

Cuadro 3.1: Relación entre la señal de referencia y los valores de la variables Var1, Var2, Var3 y Var4.

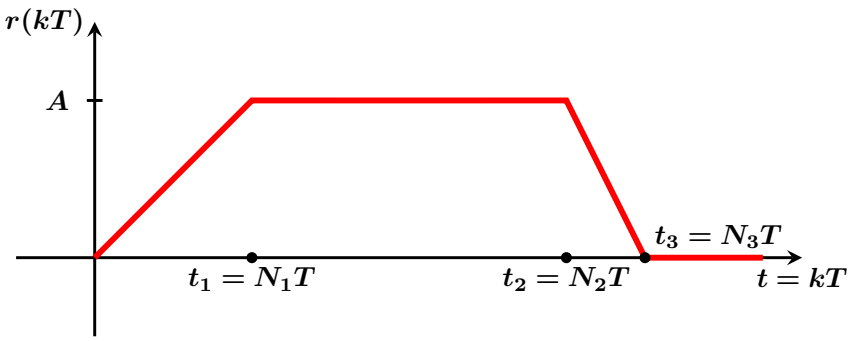

Figura 3.5: Señal de referencia trapezodidal.

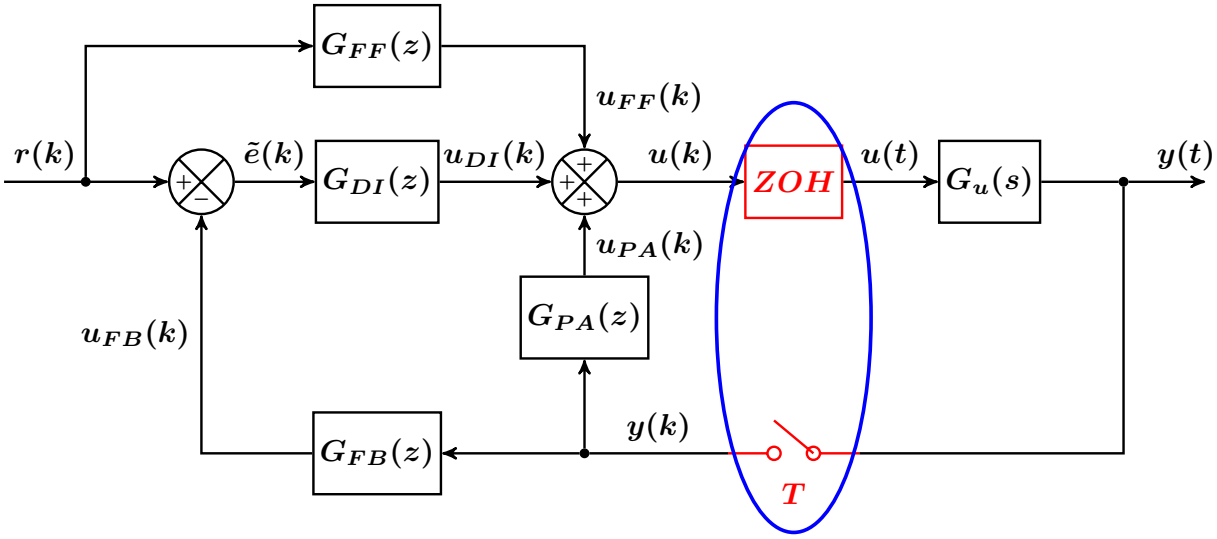

Figura 3.6: Lazos de control programados.

Primero es necesario configurar el periodo de envío de datos (por defecto a 10 ms) y el tiempo durante el cual queremos recoger dichos datos (por defecto 2 segundos).

Además, podemos configurar la aplicación para recibir las siguientes señales del motor:

- **Position**: La posición en rad, donde 0 rad es la posición inicial.
- **Speed**: La velocidad en rad/s.
- **Current**: La corriente consumida por el motor en $amperios$.
- **Reference**: La referencia $r(k)$
- **Error**: El error, $e(k) = r(k) - y(k)$
- **Error_ref_fb**: $\tilde{e}(k)$, es la diferencia entre la referencia y la salida del lazo realimentado (FB).
- **Control FeedForward**: La señal de control del lazo prealimentado en voltios ($u_{FF}(k)$).
- **Control Direct**: La señal de control del lazo directo en voltios ($u_{DI}(k)$).
- **Control FeedBack**: La señal de control del lazo realimentado en voltios ($u_{FB}(k)$).
- **Control Parallel**: La señal de control del lazo paralelo en voltios ($u_{PA}(k)$).
- **Control**: La señal de control del controlador en voltios ($u(k)$).
- **ControlSat**: La señal de control del controlador saturada a la máxima señal permitida por el motor (12 V).
- **Output**: La señal de salida ($y(k)$) con la cual ha trabajado el controlador.

Figura 3.7: Pestaña de configuración de la perturbación de la aplicación *secoStudentsApp*.

3.6.6. Botones y Leds

Existen tres botones para interaccionar con los motores.

- **Send**: Conecta la aplicación con el puesto del laboratorio seleccionado y envía la configuración de trabajo.
- **Reset**: Resetea el puesto del laboratorio en caso de haber ocurrido algún problema.
- **Disconnect**: Desconecta la aplicación del puesto de laboratorio.

Además existe una ventana de visualización que permite saber cual es el turno en el que nos encontramos y el proceso de ejecución, de tal manera que:

- **Waiting Queue:** Nos indica la posición en la cola para que nuestro trabajo entre en el puesto del laboratorio.
- **Led rojo**: Indica que hay más de un trabajo por delante del nuestro o que aún no se ha enviado ningun trabajo.
- **Led amarillo**: Indica que el próximo trabajo a ejecutarse será el nuestro.
- **Led verde**: Indica que el trabajo se está ejecutando.
- **Led azul**: Indica que la aplicación está recibiendo los datos.

3.6.7. Salvar y cargar una configuración

Para facilitar el trabajo de los alumnos, así como la evaluación, se ha implementado un menú en la barra superior de la aplicación (**File**), que permite cargar (**Load**) y salvar (**Save**) configuraciones.

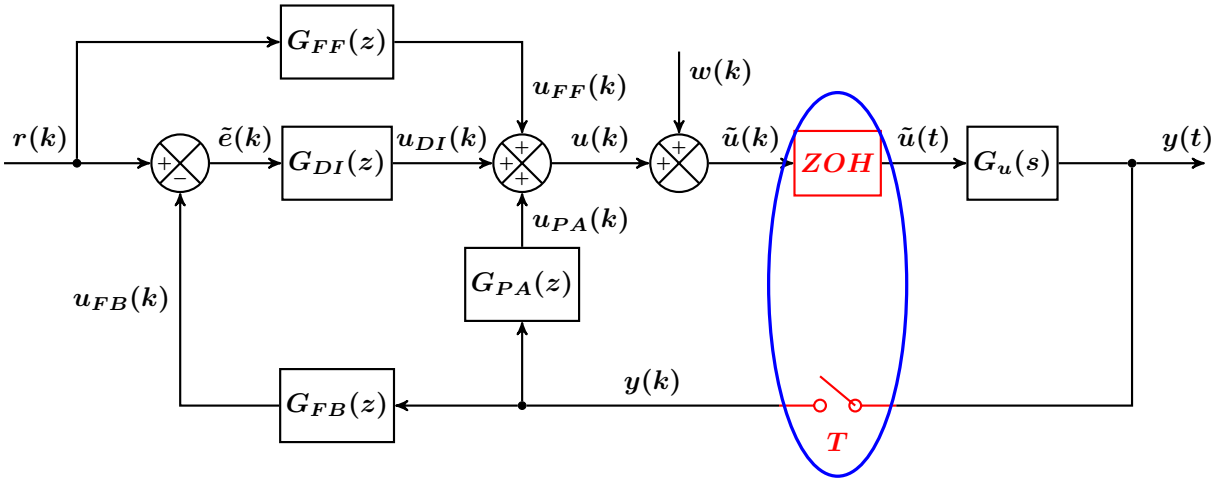

Figura 3.8: Lazos de control programados con la señal de perturbación

Para cada experimento podemos guardar la configuración, de tal manera que se pueda replicar fácilmente, tanto por el alumno a la hora de repetir el experimento, como por los profesores a la hora de evaluar.

Será imprescindible el envío de los ficheros de configuración para la evaluación de la asignatura.

3.6.8. Visualización

El segundo módulo de *secoStudentsApp* corresponde a la ventana de visualización. En dicha ventana, una vez ejecutado un controlador y pasado el tiempo de ejecución seleccionado, aparecerán las señales seleccionadas en la pestaña *Communication*.

La ventana de visualización nos permite realizar las siguientes acciones:

- Obtener las coordenadas de un punto: Click con el botón derecho.
- Zoom in/out: Rueda del ratón.
- Selección de área: Seleccionar con el botón izquierdo la esquina superior izquierda del área a hacer zoom y arrastrar hasta el punto correspondiente a la esquina inferior derecha.
- Volver al estado inicial: Presionar la tecla 'a'.

Es posible guardar las imágenes que aparecen en la ventana gráfica a través del menú `Plot` → `Save`.

Además, es posible cargar ficheros almacenados anteriormente en la ventana gráfica a través del menú `Plot`. Al presionar en `Load` aparecerá una ventana de navegación que permite seleccionar tantos ficheros como queramos.

NOTA: Es importante tener en cuenta que la aplicación gráfica sólo dibuja una gráfica por cada variable. Por lo que si se cargan por ejemplo 2 ficheros de la misma variable, la señal aparecerá distorsionada.

3.6.9. Cámera Web

La ventana de la cámara web muestra el movimiento del motor una vez que el trabajo ha entrado en cola.

3.7. Prueba de funcionamiento

En esta sección desarrollamos dos ejemplos para observar el funcionamiento de *secoStudentsApp*.

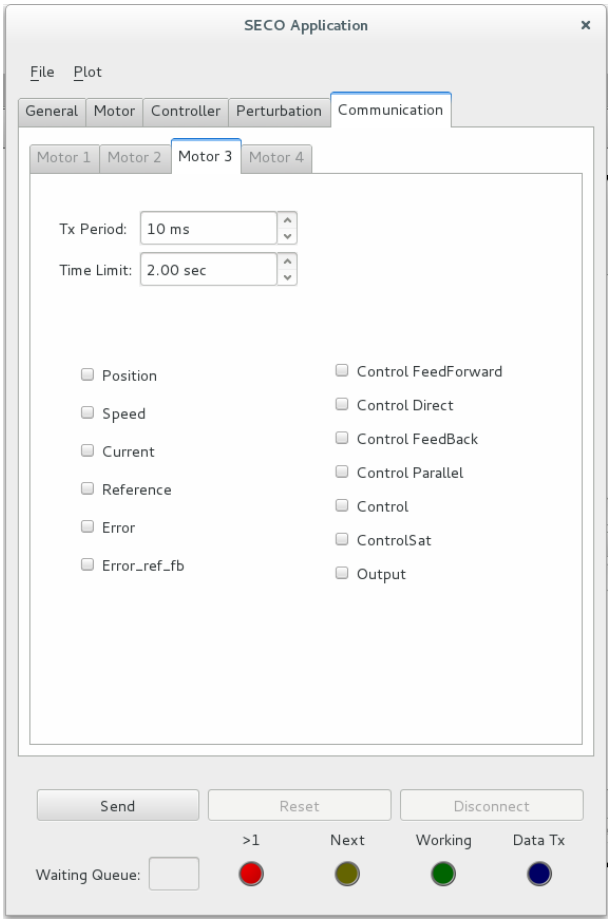

Figura 3.9: Pestaña de configuración de la Comunicación de la aplicación *secoStudentsApp*.

3.8. Ejemplo 1: Lazo abierto

En este ejemplo vamos a estudiar la respuesta en velocidad a un escalón de 10 rad/s del sistema en lazo abierto.

Seleccionaremos el directorio de trabajo, por ejemplo *data* y el prefijo de nuestros ficheros *test1*

Mantenemos nuestro Motor 3 con la frecuencia del PWM a 20kHz.

Seleccionaremos un periodo de muestreo de 5 ms, el controlador PID y la variable de control Velocidad. Seleccionaremos STEP como referencia y VAR1 a 10 rad/s. Finalmente, pondremos a 1 el valor de K_p en el lazo "FeedForward" y 0 en el resto de parámetros.

Vamos a estudiar la velocidad y la señal de referencia, por lo que marcaremos esas casillas en la pestaña *Communication*, Seleccionaremos un periodo de envío de 5 ms y 0.5 segundos de experimento.

Presionamos `Send` y una vez ejecutado el proceso deberíamos ver en la ventana de visualización el resultado del experimento, similar a la Figura 3.10.

3.9. Ejemplo 2: Controlador PID en el lazo directo

En este ejemplo vamos a estudiar la respuesta en posición a un escalón de 10 rad de un controlador P con $K_p = 10$ en el lazo directo.

Seleccionaremos el directorio de trabajo, por ejemplo *data* y el prefijo de nuestros ficheros *test2*

Seleccionaremos un periodo de muestreo de 5 ms, el controlador PID y la variable de control Posición. Seleccionaremos STEP como referencia y var1 a 10 rad. Finalmente, pondremos a 10 el valor de K_p en el lazo "Direct".

Vamos a estudiar la posición, velocidad, error, referencia, señal de control y señal de control saturada, por lo que marcaremos esas casillas en la pestaña *Communication*, manteniendo la frecuencia de envío a 10 ms y 2 segundos de experimento.

Presionamos `Send` y una vez ejecutado el proceso deberíamos ver en la ventana de visualización el resultado del experimento similar a la Figura 3.11.

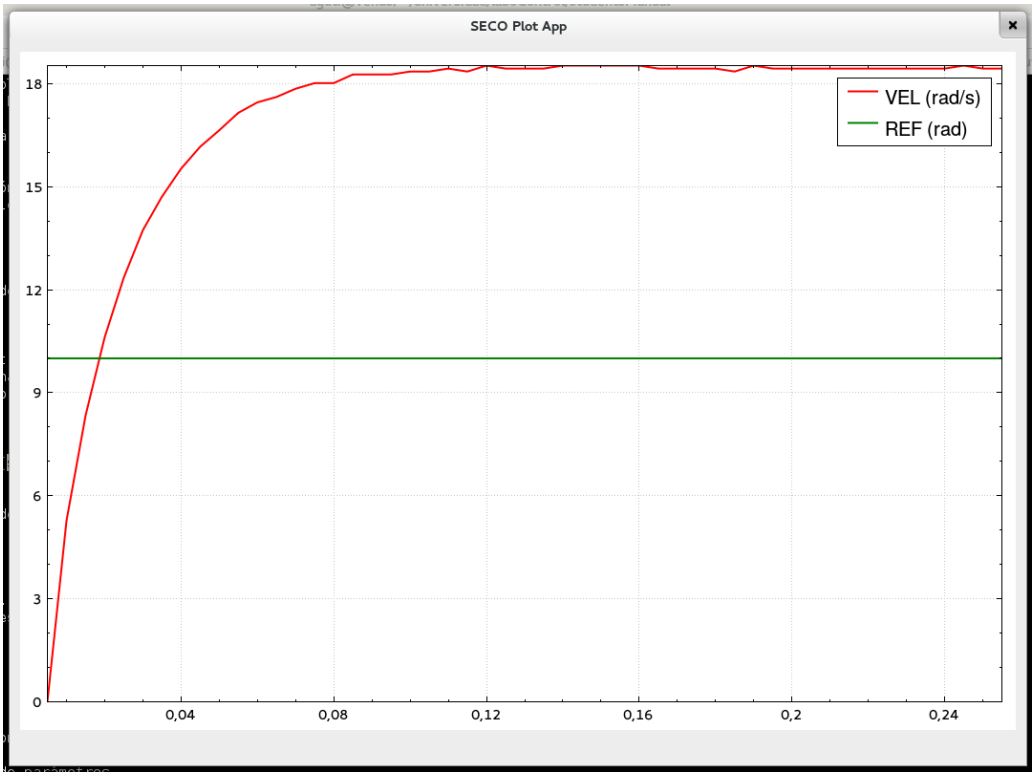

Figura 3.10: Gráfica tipo del Experimento 1.

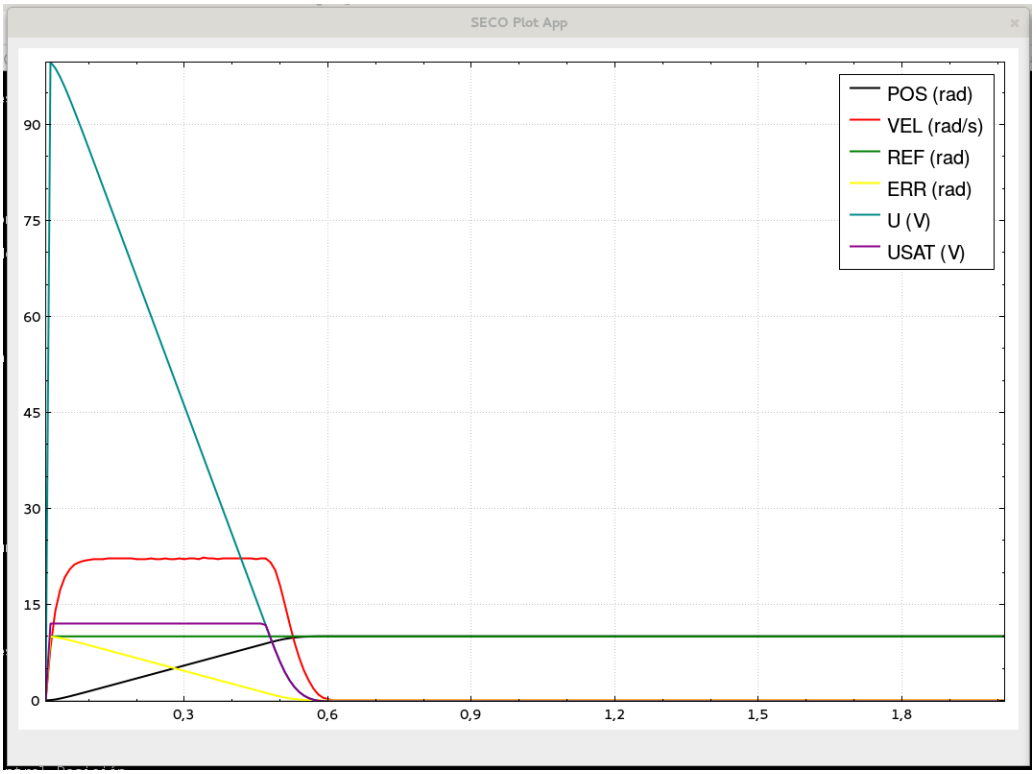

Figura 3.11: Gráfica tipo del Experimento 2.