# UNIVERSITAT JAUME I

**ESCOLA SUPERIOR DE TECNOLOGIA I CIÈNCIES EXPERIMENTALS**

**GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES**

# *DESARROLLO DEL SISTEMA DE CONTROL DE UNA PLANTA PILOTO DE LABORATORIO PARA EXPERIMENTACIÓN DE TÉCNICAS DE CONTROL DE SISTEMAS DE BOMBEO*

**TRABAJO FIN DE GRADO**

AUTOR/A

Andrés Orenga Montoliu

DIRECTOR/A

Roberto Sanchís Llopis

**Castellón, Julio de 2017**

*Quisiera agradecer a varias personas la ayuda recibida a lo largo de la carrera y durante la realización del trabajo de fin de grado:*

*En primer lugar, a mis padres Pascual y Carmen, y a mi hermana Sonia por todo el apoyo recibido durante todos estos años ya que sin ellos no habría llegado hasta aquí.*

*En segundo lugar, a esas personas que empezaron siendo mis compañeros de clase y que hoy en día los considero parte de mi familia.*

*A mi tutor Roberto Sanchís, por toda la ayuda recibida durante la realización de TFG.*

*Y finalmente, a todos mis amigos y seres queridos por aguantarme y ayudarme en todo lo necesario.*

.

# RESUMEN TFG

El proyecto consiste en el desarrollo del sistema de control de una planta piloto de laboratorio que incluirá varios depósitos interconectados, bombas, sensores de nivel, sensores de caudal y válvulas. Dicha planta será utilizada para probar esquemas de control y optimización de sistemas de bombeo.

El proyecto incluye la selección de los sensores de nivel y caudal, así como de los actuadores (bombas, válvulas y equipos de control), su prueba y puesta a punto en el laboratorio, y la selección y programación del equipo de control de la planta, que estará basado en un PLC industrial. El equipo de control de la planta deberá poder comunicarse con un ordenador a través de un servidor OPC, de forma que se puedan experimentar de forma simple estrategias de gestión óptima del sistema de bombeo programadas en el ordenador.

Una de las claves del sistema será el control en cascada del nivel y el caudal de varios depósitos interconectados. A este respecto, resulta clave la selección y prueba de su correcto funcionamiento de los sensores de nivel y caudal, las bombas con sus amplificadores, y las válvulas. La dificultad estriba en que la planta estará hecha a una escala muy pequeña respecto de las instalaciones reales, y deberá funcionar con una dinámica mucho más rápida (periodos de muestreo del orden de 1 segundo en lugar de 1 minuto), lo que puede ser un inconveniente importante a la hora de medir y controlar el caudal.

Para la realización del proyecto, va a ser necesaria la selección y posterior comprobación del correcto funcionamiento de los distintos componentes necesarios para el montaje del experimento, tales como: caudalímetros, sensores de nivel, válvulas direccionales y bombas hidráulicas. Para ello se va a necesitar realizar el montaje de un sistema de bombeo, utilizando depósitos para controlar el nivel del agua, un depósito más grande desde donde se bombeará el agua con las bombas, tubos y todas las posibles conexiones para una correcta distribución de los fluidos.

Para la comprobación de los distintos componentes, se usará un Arduino UNO R3 conectado a un PC mediante una aplicación de Java. Con esta plataforma se experimentará con el control de caudal y nivel, verificando el correcto funcionamiento de los sensores y actuadores, empezando las pruebas mediante experimentos en bucle abierto y su posterior ajuste mediante un PID, para así lograr que el bucle de control corrija eficazmente y en el mínimo tiempo los efectos de las perturbaciones, y comprobación en bucle cerrado.

Una vez comprobados los componentes, se seleccionará y programará un PLC industrial con todos los módulos de entrada/salida necesarios para controlar todo el sistema de bombeo, de forma que éste se comunique con un servidor OPC instalado en un ordenador, para que la aplicación de gestión óptima de sistemas de bombeo pueda pasarle fácilmente al PLC las consignas de caudales y niveles, y de forma que se puedan leer los valores de los distintos sensores del sistema.

# *Índice*

# *Índice de tablas*

# *Índice de figuras*

# 1. Memoria

## 1.1. Objetivo

El objetivo del Trabajo de Final de Grado consiste en el desarrollo de un sistema de control de una planta piloto de laboratorio para la experimentación de técnicas de control de sistemas de bombeo, incluyendo el diseño de las conexiones entre los sistemas de depósitos para que se puedan configurar como un control multivariable.

Para todo ello se tendrá que definir y programar las distintas conexiones eléctricas y la electrónica, para la obtención de datos, tales como el nivel del líquido de los depósitos y el caudal suministrado encada momento, y la de potencia para atacar a las distintas bombas utilizadas para el llenado y vaciado de los depósitos, y el uso de distintos softwares necesarios para el uso mediante el PC, del control de los mismos. Además, la elección de los distintos componentes necesarios para la realización del mismo.

## 1.2. Antecedentes

Para la realización de este proyecto, se dispone en el laboratorio de investigación de una preinstalación de unos pequeños sistemas de bombeo. Lo que se pretende realizar a partir del mismo es un sistema de control piloto para poder experimentar la gestión y optimización de sistemas de bombeo. Se realizará un diseño de la misma, la cual puede añadir y quitar componentes para poder ampliar o reducir las operaciones realizadas para el control de bombeo de manera que sea un sistema multivariable y esté controlado con la ayuda de sensores y actuadores.

En el laboratorio, se dispone de en un sistema de control automático de una planta de laboratorio basada en dos depósitos. Basada en dos sistemas iguales, formados por un deposito inferior de acumulación, desde donde se proporciona agua hacia unos depósitos superiores mediante una bomba de agua. Los depósitos superiores controlan el nivel de agua mediante dos sensores: uno capacitivo de nivel y otro de ultrasonidos, colocados en la parte superior de cada uno de los depósitos.

La idea en este caso será, mediante la utilización de los depósitos, probar esquemas de control y la optimización de sistemas de bombeo mucho más grandes que el que se va a utilizar. En los que se deberán modificar los algoritmos de control del sistema y la reconfiguración del mismo.

Por otra parte, se tendrán que buscar y comprar los distintos componentes que se utilizaran para el montaje del mismo, ya que los sensores capacitivos y los caudalímetros disponibles no funcionan correctamente. Para ello se buscarán unos sensores capacitivos nuevos, bombas sumergibles, caudalímetros y válvulas direccionales. Todo ello, dentro de un coste que no suponga ser muy elevado.

## 1.3. Requisitos de diseño

Los requisitos para este diseño, se elegirán a partir de la planta que se encuentra en el laboratorio. La planta de la que se dispone, es una planta ya diseñada a escala, por lo que los componentes a utilizar, se deberán acoplar tanto al nivel de agua de los depósitos y a los caudales trasegados necesarios. Lo que se intenta buscar en este proyecto, es que este pueda permitir una lectura constante del caudal de agua de entrada, como en la salida, pudiendo variar el camino del caudal de agua en todo momento a partir de válvulas. Además, también se deberá medir en todo momento el nivel de agua que se encuentra en los depósitos. Todo ello se deberá ajustar para que tenga un buen funcionamiento y que el coste total de toda la instalación sea lo más económica posible, pero con buenos componentes y que tengan una buena respuesta.

Los requisitos que debe cumplir el proyecto, son los siguientes:

- El software para poder implementar los algoritmos de control deberá ejecutarse mediante un PC para que este se comunique con la plataforma de adquisición.

- Se precisará de unos componentes que nos permita obtener una lectura y control del nivel del agua que se encuentra en el depósito en cada momento. Los depósitos que se utilizaran no pueden albergar un volumen elevado, por lo que es necesario que este componente se adapte al depósito que se va a utilizar. En el laboratorio, se pueden encontrar distintas soluciones al mismo, desde sensores capacitivos de nivel, hasta sensores de ultrasonidos. Todos ellos se deberán probar y en el caso de que no funcionarán correctamente, adquirir unos nuevos para poder realizar el control de nivel.

- Para poder saber la cantidad de agua que está entrando dentro de los depósitos en cada momento, se precisara de un componente que permita la lectura de esa cantidad de agua que circula. Los componentes para poder realizar esas medidas, son los caudalímetros, de los que más adelante barajaremos las distintas clases que ofrece el mercado, y cuál de ellas se adaptaría mejor a nuestra instalación. Teniendo siempre en cuenta que deben tener una baja caída de presión.

- El componente a utilizar para poder suministrar el agua a los depósitos, es una bomba de agua. El mercado ofrece distintos tipos de bombas, desde bombas sumergibles de continua, hasta bombas trifásicas de una mayor potencia. Pero como en este caso, ya se dispone de una planta a escala, se deberá encontrar una bomba que pueda ser útil en esta planta dependiendo el nivel y el caudal trasegado. Además, analizaran las distintas opciones que se acoplan mejor a nuestra instalación.

- El montaje eléctrico de los todos los componentes, así como su cableado y las conexiones necesarias para su correcto funcionamiento. El montaje del sistema de control mediante la unión de los distintos componentes como caudalímetros, y bombas, unidos entre ellos a través de unos tubos de plástico termoestable por los que circulara el agua. A su vez, el montaje de las entradas y salidas de agua al depósito y su distribución, se realizará con la ayuda de tubos de PVC termoplásticos y válvulas direccionales todo/nada.

- Los depósitos de agua y los de acumulación, se encuentran preinstalados en el laboratorio. Los depósitos son de 25 litros cada uno, más que suficiente para poder realizar el estudio, y los de acumulación tienen una capacidad más que suficiente, además de estar conectados entre ellos para que siempre haya agua en los dos depósitos, también se recirculara el agua para que la salida de directamente a los depósitos de acumulación.

## 1.4. Análisis de soluciones

El proyecto que vamos a realizar es en pequeña escala, por lo que se deberán buscar componentes eficaces y que puedan suministrar un caudal de agua bajo, que los sensores permitan la instalación dentro del depósito que ya se dispone en el laboratorio y que los componentes tengan una baja caída de presión. Todas las comprobaciones de los materiales y los bucles de control, se realizarán a partir del Arduino.

### 1.4.1. Adquisición de datos

Para la adquisición de datos, el mercado ofrece varias posibilidades. Desde la utilización de tarjetas de adquisición de datos, las cuales se integran en el ordenador, hasta microcontroladores o placas basada en microcontroladores.

El proyecto se centra en el uso de un PLC. El PLC (controlador lógico programable) o autómata programable. Es una computadora utilizada en la ingeniería automática o industrial, para automatizar procesos electromecánicos. Los PLC son utilizados en muchas industrias y máquinas. A diferencia de las computadoras de uso general, el PLC está diseñado para múltiples señales de entrada y de salida, rangos de temperatura ampliados, inmunidad al ruido eléctrico y resistencia a la vibración y al impacto. Un PLC es un ejemplo de un sistema de tiempo real, donde los resultados de salida deben ser producidos en respuesta a las condiciones de entrada. Dentro de las ventajas que estos equipos que se poseen se encuentra que, gracias a ellos, es posible ahorrar tiempo en la elaboración de proyectos, pudiendo realizar modificaciones sin costes adicionales. Por otra parte, son de tamaño reducido y con un bajo coste de mantenimiento, además permiten ahorrar dinero en mano de obra y la posibilidad de controlar más de una máquina con el mismo equipo.

Además, para poder comprobar el correcto funcionamiento de los distintos componentes, se utilizará una tarjeta con microcontrolador de la familia Arduino, con el fin de realizar todas las pruebas pertinentes antes de basarse en el PLC. Las placas de Arduino son comúnmente usadas en la parte de sistemas de control para la realización de prácticas o proyectos por parte de la universidad. La universidad también dispone de las tarjetas de la familia Microchip (PIC y dsPIC).

Para ello explicará brevemente y en términos sencillos que es microcontrolador.

Un microcontrolador es un dispositivo que se puede programar para hacer distintas tareas y para ello requiere de la ayuda de ciertos periféricos, pero también necesita de otros elementos adicionales para poder interactuar con él. Es como si se tuviera la CPU de un ordenador, la cual es capaz de realizar muchas operaciones, pero sin periféricos donde poder visualizar la información no se podría hacer un uso adecuado del mismo.

Arduino es una plataforma de código abierto con la cual se pueden crear proyectos de electrónica digital, domótica, automática, etc. De forma rápida sencilla y económica. En la red se pueden encontrar muchos proyectos ya creados con la plataforma de Arduino. Las placas más usadas son Arduino UNO Genuino y Arduino MEGA, basadas en microcontroladores ATmega320 y ATmega2560.


La principal diferencia entre Arduino y PIC, es que Arduino es una placa de desarrollo basada en microcontrolador y PIC es una familia de microcontroladores. En el mercado existen muchas placas de desarrollo basadas en microcontroladores PIC, pero son mucho más caras que las placas Arduino.

Las placas de desarrollo Arduino son de código abierto, tanto a nivel de hardware como de software. En Microchip, en cambio se debe pagar una licencia para poder usar su software. También existe la posibilidad de fabricarse uno mismo su propio Arduino, ya que no se infringen los derechos de autor. El software de uso para las placas Arduino, se puede conseguir gratuitamente desde la página oficial de Arduino.

Arduino es bastante útil para las personas aficionadas a la electrónica o para estudiantes que desean introducirse en el mundo de la programación. Y es que, para hacer una aplicación en Arduino, solo basta con comprar cualquiera de sus placas, descargar el software de programación y en cuestión de tiempo, estará lista la primera aplicación (para ello son necesarias nociones de programación en leguaje C). Además, es práctica, ya que para una persona que no sea experta en la utilización de microcontroladores, con Arduino, podría diseñar, por ejemplo, un sistema de iluminación LED.

Por tanto, Arduino es más fácil de aprender que PIC, y como es bastante económico, se utilizará Arduino para realizar las pruebas iniciales en el proyecto, ya que diseñar una placa de desarrollo o un microcontrolador, sería mucho más costoso que si se compra directamente el Arduino. Además, esto nos permite tener más flexibilidad para posibles cambios de sensores o bombas, que, si se diseñara una placa, no permitiría realizar cambios, y no tendría en cuenta todos los posibles sensores que se pueden conectar en el futuro.

En Arduino existen dos maneras distintas de comunicarse con el ordenador. Por el puerto USB o serie, mediante WiFi o por cable Ethernet. La manera más económica de comunicarse con el Arduino, es mediante el cable USB, además de proporcionar al Arduino alimentación eléctrica sin la necesidad de utilizar una fuente de alimentación externa, también nos permite cargar el programa dentro del Arduino mediante la aplicación de software Arduino. En el caso de ser mediante WiFi o Ethernet, además de encarecer el producto, se necesita de una fuente de alimentación externa para su funcionamiento.


En cuanto al software a utilizar, las opciones más comunes son la utilización de Matlab u Octave. En nuestro caso, utilizaremos directamente el software de Arduino para escribir y compilar el código, ya que parte del código a utilizar, ha sido usado en algunas de las prácticas realizadas durante la carrera. En nuestro caso, realizando algunas modificaciones del mismo, podemos ajustarlo para que funcione todo correctamente, e incluso agregar parte del código para poder utilizar más sensores en nuestro proyecto.

Lo que se dispone a hacer con el Arduino, son las distintas comprobaciones de los distintos componentes que componen el proyecto, así como su correcto funcionamiento y además el diseño de los bucles de control de los distintos componentes.

### 1.4.2. Bomba

Una de las partes importantes, el componente que hará que circule el agua por la instalación, son las bombas. El mercado ofrece múltiples bombas de agua, como, por ejemplo, bombas de agua trifásicas, bombas de inducción e incluso bombas fotovoltaicas. En nuestro caso, se precisa de una bomba de agua sumergible y que pueda bombear agua entre 0-10 l/min a una altura de 0.5m (0.05bar). Por ello, después de buscar en las distintas páginas y catálogos, se decide obtener una bomba de agua sumergible de continua. En un principio en el laboratorio se dispone de unas bombas de 12V, unas bombas que, para trabajar con el Arduino y realizar las pruebas iniciales de componente, funcionan perfectamente, pero no ofrecen un nivel de caudal necesario para la realización del proyecto. Como las bombas serán alimentadas directamente por el PLC, se necesitarán bombas de 24V. Entonces se decide obtener esta bomba de agua: Bomba sumergible 24V “Comet Elegant”, de la tienda online *fotovoltaica*. Estas bombas funcionan con un caudal máximo de 10 l/min a desnivel cero. Unas bombas que pueden llegar a trabajar hasta los 5 metros altura, pero como en nuestra instalación, la presión máxima sería de unos 0.05 bar, la bomba cumpliría con las expectativas y no perdería apenas parte de su caudal nominal. Además, estas bombas, tan solo consumen 1A como máximo, por lo que las bombas se adaptan perfectamente a la solución adoptada. Para poder controlar las bombas, se precisará de un controlador de motores de continua, que más adelante barajaremos las distintas opciones que existen en el mercado.

*Figura 1. Bomba Comet Elegant 24V*

### 1.4.3. Sensor de nivel

A la hora de empezar a comprobar los distintos componentes (ya que no se disponen aún de todos los componentes necesarios para la realización del proyecto), se usó un sensor de nivel capacitivo, en el que obteníamos sus datos mediante una salida analógica al Arduino entre 0-10V. En Arduino solo se pueden utilizar tensiones menores o iguales a 5V (se podría averiar el Arduino), por lo que, con la ayuda de un divisor resistivo con dos resistencias iguales, convertiremos la tensión de salida a 0-5V. En el mercado podemos encontrar distintas opciones como los sensores de ultrasonidos, laser e incluso capacitivos como los que se disponen en el laboratorio. Más adelante de barajarán las distintas opciones que existen y se verá que, para poder realizar un buen montaje de la instalación, el mejor sensor que se puede utilizar es un sensor de nivel capacitivo como los que se encuentran en el laboratorio. La decisión de obtener un nuevo sensor de nivel fue debida a que los que se encuentran en el laboratorio para su uso en prácticas, no realizan correctamente su función y tienen un tiempo de respuesta muy lento. Se propuso buscar un sensor de nivel que no fuera capacitivo, pero los sensores de láser y ultrasonidos con un buen funcionamiento son muy caros, por lo que encarecería el proyecto. También se probó un sensor de ultrasonidos muy económico, el cual media la distancia a la que se encuentra la parte superior del nivel del agua, pero no era muy preciso y a su vez muy lento. Este sensor está especialmente diseñado para trabajar con el Arduino, pero como al final del proyecto este no aportaba la precisión de medida requerida, se optó por el uso de un nuevo sensor de nivel capacitivo de mayor calidad. Tras la búsqueda del mismo, se decidió por el sensor sonda

capacitiva de nivel continua de líquidos “Liquicap T FMI21”, de la tienda eDirect, el cual trabaja con una tensión entre 10-30V y con un consumo de corriente menor a 22mA. Este en concreto, nos da la información mediante una salida de corriente analógica entre 4-20 mA, la cual convertiremos en tensión con la ayuda de una resistencia. Este sensor de nivel se ajusta perfectamente a lo que se estaba buscando. Además, a la hora de comprar el sensor, da la posibilidad de pedir las varillas del sensor en la dimensión que se precise, por lo que se escogieron unas varillas de 50cm, las cuales serían perfectas para encajar dentro del depósito de agua.

*Figura 2. Sensor de nivel capacitivo.*

### 1.4.4. Caudalímetro

Para poder saber la cantidad de caudal que está circulando en la instalación y poder realizar un control del mismo, se usara unos caudalímetros. En el laboratorio existen unos caudalímetros de paletas, los cuales no funcionan correctamente y tienen un tiempo de respuesta muy alto, por lo que se opta por elegir otro tipo de caudalímetro para realizar el proyecto. En el mercado se pueden encontrar numerosas opciones de tipos de caudalímetros que puedan realizar la función que nosotros deseamos para el proyecto. Pero en nuestro caso, se necesita un caudalimetro que permita entregar los datos obtenidos mediante una señal analógica, que además permita realizar una buena lectura de caudal entre 0-10 l/min, que tenga una caída de presión muy baja y que tuviera el menor tiempo de respuesta posible. Para el modelo de caudalimetro que estábamos buscando no existen en el mercado numerosas soluciones, ya que la mayoría de ellos están pensados para un caudal mucho mayor del que se va a usar en el proyecto. Al final se decidió adquirir el caudalímetro de ultrasonidos “Cynergy”, de la tienda cinergy3, el cual tiene un rango de medida entre 0.4-8 l/min, un caudal más que suficiente para poder realizar el control del sistema de bombeo. Además, para poder obtener los datos del mismo, se obtiene la información por medio de una salida de tensión analógica entre 0V-5V, una tensión correcta para no estropear el Arduino ni el PLC.


*Figura 3. Caudalímetro ultrasonidos.*

### 1.4.5. Amplificadores

Una vez decidida la familia de microcontroladores y las bombas que se van a utilizar para la realización del proyecto, se buscará un controlador de motores de continua (Amplificadores), para el control de las mismas, ya que las bombas necesitan una tensión de 12V en las pruebas iniciales y 24V las bombas para la resolución final. Ambas dos consumen 2A cada una, por lo que los amplificadores, deberán soportar una tensión mayor a 24V y una corriente superior a 2A que es lo que consume cada bomba.

La corriente máxima que soporta el Arduino es de 2A y su tensión máxima es de 12V, por lo que las bombas no se pueden conectar directamente al Arduino. Así pues, se alimentarán con la ayuda de una fuente de alimentación externa y controladas por el amplificador, para así poder regular la tensión de salida en las bombas. Ademas, estos controladores deberán tener una entrada analogica de tensión para poder ser controlados mediantes el Arduino o el PLC.

Tras buscar los distintos tipos de amplificadores que hay en el mercado, se decide optar por el siguiente controlador de motores de DC “United Automation DMC-24-40”, de la tienda Farnell. Este permite controlar motores de continua en un rango de tensión entre 6-24V y con una corriente máxima de 40 A, más que suficiente para las bombas que vamos a utilizar, con el cual controlaremos la tensión de las bombas mediante una entrada analógica desde el Arduino entre 0-5V.


*Figura 4. Controlador motor DC (Amplificador).*

### 1.4.6. Electroválvula

La electroválvula elegida para la realización del proyecto será en concreto un todo/nada, su funcionamiento consiste en abrir y cerrarse gracias a una señal de tensión, cuando a esta le llegan 24V, se abre y cuando no, se cierra. Además, esta electroválvula permite poder abrirse y cerrarse manualmente en el caso que se desee. A partir de ella modificaremos la circulación del caudal de agua en la instalación, para así poder dirigirla a cada uno de los distintos depósitos. Para que la solución no encareciera el presupuesto del proyecto, se opta por una electroválvula todo/nada de riego

*Figura 5. Electroválvula de riego*

### 1.4.7. PLC

Como último componente, se utilizará un PLC que se encuentra en el laboratorio, de la marca WAGO, en concreto el modelo 750-841, que se puede configurar a través de un cable RJ-45 o mediante una entrada USB, con su software, CODESYS V2.3. A este PLC se le pueden añadir los módulos de entradas y salidas en función del uso que se le vaya a dar. En nuestro caso se necesitarían 3 entradas analógicas, 2 salidas analógicas y 2 salidas digitales. Por lo que se necesitará:

- Módulo de expansión 750-501: con dos salidas digitales de 24VDC y 0.5A, que se usarán para la conexión de las electroválvulas.
- Módulo de expansión 750-468: con cuatro entradas analógicas entre 0-10VDC, que se usaran para la conexión de los caudalímetros y el sensor de nivel.
- Módulo de expansión 750-550: con 2 salidas analógicas de 0-10V, los cuales se usarán para el control de los amplificadores de continua.
- Módulo de expansión 750-600: se utilizada como módulo final.

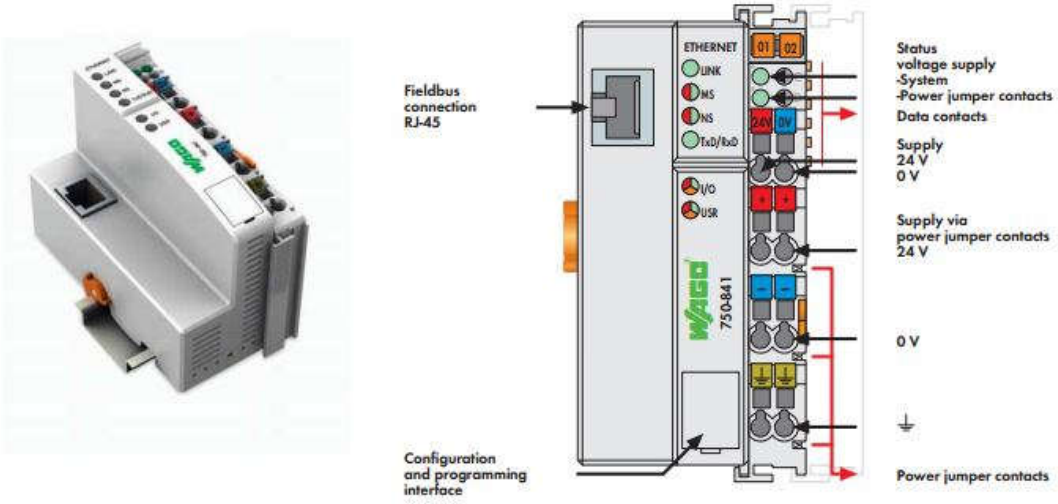



*Figura 6. PLC 740-841 WAGO*

### 1.4.8. Componentes elegidos

Los componentes necesarios para la realización del proyecto serán los siguientes:

- Bombas de 24V DC
- Sensor de nivel capacitivo
- Controladores de motores de continua (Amplificadores)
- Caudalímetros
- PLC WAGO
- Electroválvula todo/nada

## 1.5. Descripción general de la solución adoptada

Una vez analizadas las distintas opciones con las que podíamos configurar el proyecto, a continuación, se describe la solución final.

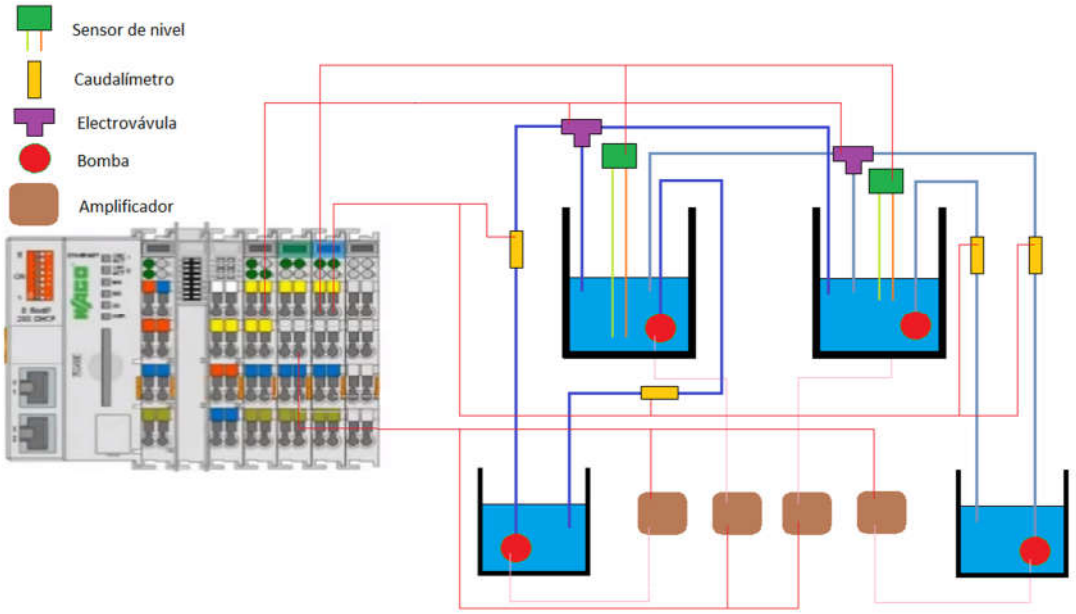



*Figura 7. Diseño instalación*

En cuanto a la instalación de bombeo, se utilizará una base de pruebas de laboratorio que se suele utilizar en algunas de las prácticas realizadas en la universidad. Se realizarán algunos cambios para poder colocar los distintos componentes que componen el proyecto y posteriormente realizar cualquier modificación pertinente. La instalación consta de dos depósitos de acumulación de agua y de dos depósitos desde donde se obtendrá el agua para realizar el experimento. En concreto, los dos depósitos de acumulación están comunicados entre sí, y los depósitos de los que se obtiene el agua, también, ya que así se asegura que siempre haya agua en

alguno de los dos depósitos para que la bomba pueda funcionar correctamente y no se estropee, ya que si la bomba está en funcionamiento mientras no hay agua, esta se averiaría. Además, la recirculación del agua usada en el análisis irá directamente a los depósitos de acumulación. Estas uniones a los depósitos traen consigo una unión mediante una válvula manual, que se cambiará por una válvula proporcional todo/nada, que será controlada mediante el PLC o manualmente, así en cualquier momento se podrá controlar a cuál de los dos depósitos irá el agua.

En la parte de las bombas, estas serán alimentadas por medio del controlador de motores de continua (amplificador), mediante una señal analógica entre 0V y 5V. La tensión suministrada a los amplificadores, se obtendrá mediante una fuente de alimentación externa, la cual utilizarán 12V para la bomba con la que se realizan las pruebas y 24V para las bombas con las que se realizarán las pruebas finales. Las bombas irán tanto a la parte superior de los depósitos como a a la inferior, con el fin de poder controlar tanto el agua que entra a la instalación como la salida de la misma. La unión de los distintos componentes se realizaría de la siguiente manera:

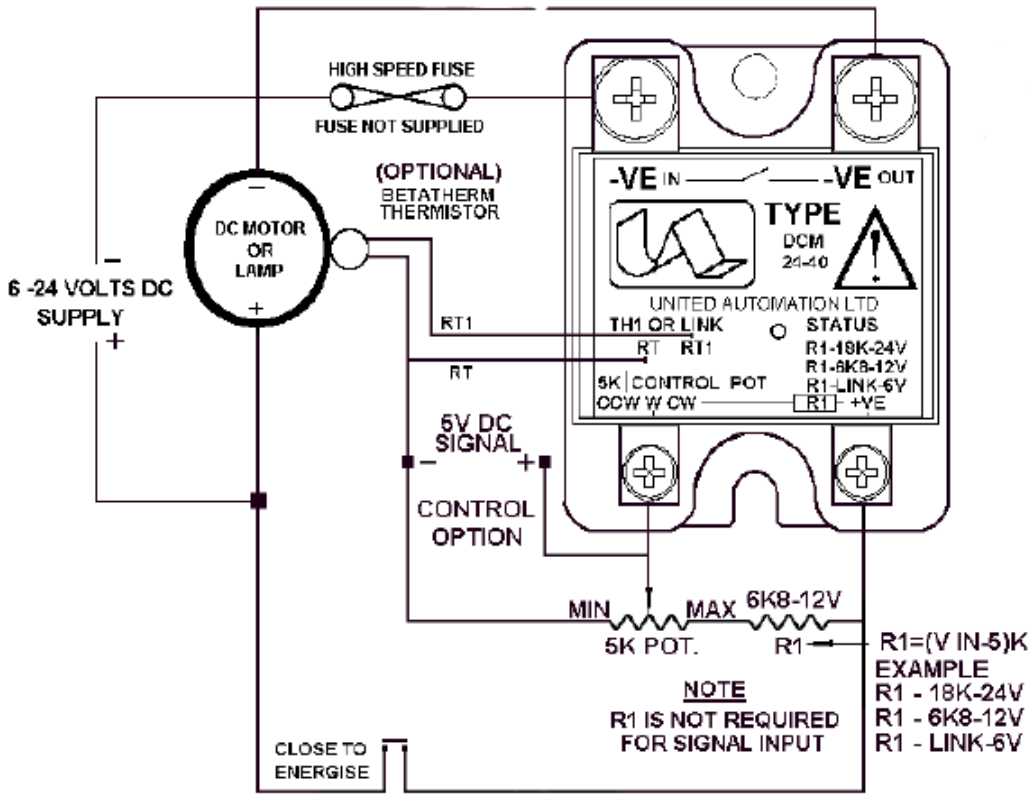



*Figura 8. Modo conexión amplificador de bombas DC*

Una vez realizadas todas las conexiones, para poder controlarlo mediante una señal analógica con el Arduino o con el PLC, se deberá conectar mediante la señal de control DC 5V al positivo y a GND a la conexión RT del amplificador.

Para la conexión del caudalimetro de ultrasonidos, este se conectará mediante unas gomas de un plástico termoestable en serie con las bombas, y de ahí ira directo al depósito. Para la alimentación del mismo, se conectará directamente a una fuente de alimentación externa, ya que el Arduino no pude alimentar directamente al caudalímetro. Lo que sí que conectaremos al Arduino serán los cables para poder obtener en todo momento el caudal que está circulando por la instalación, por medio de una señal analógica de 0V a 5V.

Por otro lado, para la conexión del sensor de nivel capacitivo, este se colocará en la parte superior del depósito mediante una unión roscada, para así poder realizar las mediciones de manera más precisa, sin que este pueda llegar a moverse. Al inicio del proyecto se usó un sensor capacitivo que se encontraba en el laboratorio. Este en concreto, necesitaba una tensión de unos 15V para que funcionara correctamente. Uno de los inconvenientes que este tenía, era que para la adquisión de datos este daba una señal analógica entre 0V-10V, por lo que para que no llegara a estropear al Arduino a la hora de conectarlo en él, se debía realizar un divisor resistivo entre la conexión con el fin de que la tensión que llegara al Arduino fuese entre 0V-5V.

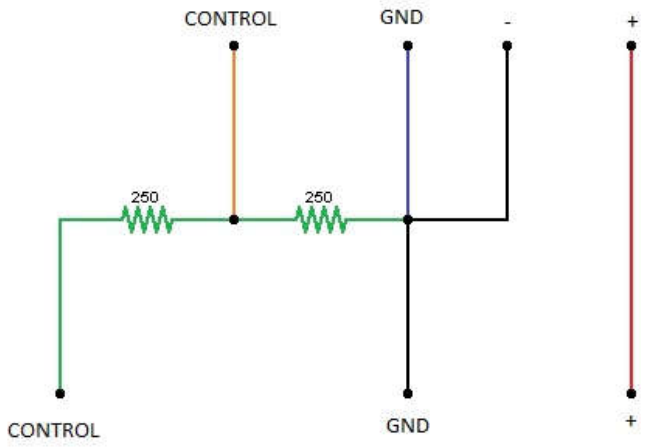



*Figura 9. Divisor resistivo*

El conexionado de las electroválvulas todo/nada es bastante sencillo, ya que estas en concreto funcionan a 24V, por lo que de normal está cerrada, y al recibir una tensión de 24V, esta

se abrirá dejando paso a que el agua sea bombeada en la dirección en la que queremos que circule el caudal.

En el inicio del proyecto, la parte de la adquisición de datos se hará con el Arduino UNO R3. El Arduino se conectará al ordenador mediante una entrada USB, por la cual recibirá los datos el ordenador que este ha obtenido de los distintos sensores y actuadores. Todo esto será controlado por una aplicación de Java usada, a lo largo de las distintas prácticas cursadas durante el grado de Tecnologías Industriales, para poder obtener los datos mediante gráficas, pudiendo variar los valores del ajuste de PID. Más adelante todo va a ser controlado mediante el PLC de WAGO. El conexionado realizado para el PLC de la compañía WAGO, se realizará a través de la ayuda de los manuales del mismo, ya que cada parte de los modelos llevan consigo unas conexiones internas distintas, por lo que dependiendo si se trata de un módulo analógico o digital, este deberá ser conectado de forma distinta. Por lo tanto, que las conexiones realizadas se harán de la siguiente manera:

## 1.6. Realización del montaje, comprobaciones, bucles de control y resultados obtenidos

Una vez elegidos todos los componentes necesarios para la realización del proyecto y como se realizarían las conexiones de cada uno de los componentes que lo forman, se dispondrá a montar y realizar todos los ajustes necesarios para que todo funcione correctamente.

Antes de empezar a realizar las comprobaciones, se deberá diseñar un código para Arduino con el que, gracias a sus entradas y salidas de señales, consiga obtener los datos a partir de los actuadores y sensores que forman la instalación. Además, este código nos permitirá realizar las distintas comprobaciones que realizaremos más adelante y con las que además obtendremos los bucles de control del caudalímetro y del sensor de nivel. Se utilizará para el control de una aplicación creada con el fin de transformar los datos obtenidos en gráficas. La aplicación **PID_Arduino** permite realizar ensayos en bucle abierto y en bucle cerrado, además de poder realizar una grabación durante el ensayo y de la misma grabación crear un fichero .txt con todos los datos recopilados durante la grabación, para posteriormente introducirlos en una aplicación de Java en la que se realizará un ensayo de escalón con el fin de obtener la curva de transferencia del mismo, y a su vez, realizar el diseño de un PID óptimo a partir de otra aplicación Java, que se utilizará en el proyecto.

### 1.6.1. Obtención de datos

Para empezar, se necesitará de un ordenador para poder realizar la búsqueda de los nuevos componentes, información y poder utilizar ciertos programas Java, que ya han sido usados anteriormente en alguna de las prácticas de la universidad, así como una aplicación que se conectará para recibir datos a través del Arduino, que también ha sido usada, con la cual se obtendrán los resultados acerca del caudal y nivel de agua mediante gráficas. A partir de la aplicación **PID_Arduino**, realizaremos todas las comprobaciones para el correcto funcionamiento de los componentes y, además, la aplicación para él, permite realizar tanto ensayos en bucle cerrado como en bucle abierto, permitiendo el diseño de bucles de control para el correcto funcionamiento entre las bombas, caudalímetros y el sensor de nivel. La aplicación permite modificar dentro de ella los valores del proporcional, integrador y derivador que componen un PID. Además de poder añadir un PID, obtenido previamente en los programas Java que nos permiten realizar un ensayo de escalón y posteriormente obtener un PID óptimo, o realizarlo manualmente. Para que el Arduino se comunique con la aplicación y pueda obtener los datos, se ha de diseñar un código que lo permita. Este código se encontrará en los Anexos.

### 1.6.2. Modificación de la estructura

A la hora de realizar las primeras pruebas, se deben realizar algunas modificaciones a la estructura inicial que hay en el laboratorio. Para ello precisaremos de unos tubos de PVC termoestables para poder realizar las distintas uniones entre las bombas y los caudalímetros, y posteriormente, conectarlos a los depósitos. La unión es muy sencilla, la bomba estará colocada en la base de unos de los depósitos de acumulación, de manera que siempre este cubierto de agua, ya que, si la bomba trabaja sin bombear agua, esta se estropearía. Con la ayuda de unas bridas, apretaremos la unión de los distintos componentes con las gomas, de manera que no haya pérdidas de agua en las uniones. Para las comprobaciones, la salida del agua, estará controlada manualmente por una válvula.

### 1.6.3. Comprobaciones

Las primeras comprobaciones, serán para poder comprobar que todos los elementos con los que vamos a realizar el proyecto, funcionan correctamente, y en el caso de que no, se obtendrán nuevos materiales para la realización del mismo. Dichas comprobaciones, se realizarán mediante un Arduino y que, además, posteriormente realizaremos con él los bucles de control del caudalímetro y el sensor de nivel. Para empezar, se realizarán con una bomba de 12V, ya que las bombas de 24V no se tienen desde el principio, con el fin de probar el caudalimetro que se

encuentra en el laboratorio y posteriormente el que se ha comprado. Una vez comprobado que los caudalímetros de paletas que se encuentras en el laboratorio son muy poco precisos, se decide empezar a realizar las pruebas con el sensor de nivel de ultrasonidos. Se observa que este funciona correctamente. El siguiente componente que se desea probar es el sensor de nivel, el cual también se comprobara con uno que se encuentra en el laboratorio, con el fin de saber si este funciona correctamente o si es necesario obtener un nuevo sensor de nivel para poder realizar el proyecto. De esta manera, se empezará comprobando el correcto funcionamiento de las bombas, por lo que realizaremos las conexiones pertinentes con el fin de que la bomba funcione correctamente.

#### 1.6.3.1. Comprobaciones en las bombas

Las bombas que se han comprado para hacer las pruebas, son muy económicas, estas no traen consigo una curva de caudal, por lo que con la ayuda de un multímetro y con el caudalímetro, se obtendrá la curva característica de la bomba de agua. Para ello, se procede a tomar distintas medidas entre el voltaje que está consumiendo la bomba y la relación caudal que está circulando por el caudalímetro. Para poder observar el caudal que está pasando por la instalación, se usará la aplicación **PID_Arduino**, que muestra las gráficas obtenidas por el Arduino a través de las señales analógicas obtenidas por los sensores y actuadores. El resultado obtenido de la curva de caudal en relación al voltaje de salida y el caudal de entrada, es el siguiente:

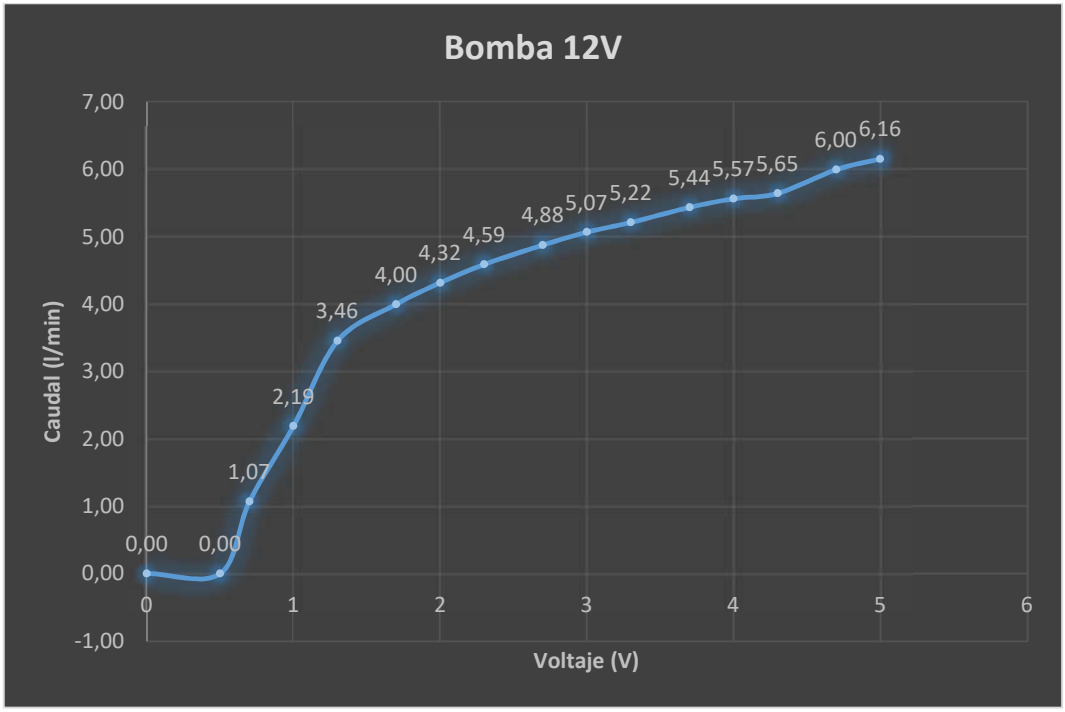



Tabla 1. Curva caudal Bomba 12V

### 1.6.3.2. Comprobaciones y bucle de control en el caudalímetro

A la hora de realizar el estudio en las bombas, previamente se ha instalado el caudalímetro en serie con la ella, con el fin de que este nos marque el caudal que está circulando por la instalación y que el caudalímetro funciona de la manera esperada. Aunque el tiempo de respuesta del mismo no pude ser menor a 250ms, ya que, si disminuimos ese periodo, el caudalímetro no funciona correctamente a la hora de entregar los datos a la aplicación. A continuación, se realizará el diseño de un PID para que la acción de control, que es el voltaje suministrado a la bomba, y la referencia, que en este caso es el caudal de agua que está circulando por el caudalímetro, tenga la mínima desviación o error entre el valor medido y el valor deseado. Para ello, con la ayuda de unas herramientas Java, utilizadas a lo largo del grado de Tecnologías Industriales, se ajusta el PID óptimo. Las herramientas utilizadas para poder realizar el diseño del PID óptimo son las siguiente:

- **Ejs_model_identescalon24**: La función de esta herramienta consiste en obtener la función de transferencia a partir de un ensayo de escalón. Este ensayo se realiza previamente con la aplicación **PID_Arduino**. Una vez obtenidos los datos, este programa obtiene la curva de transferencia del mismo con un error mínimo.

- **Ejs_model_PIDFREC2053**: La función de esta herramienta es, a partir de una curva de transferencia, obtenida anteriormente por el programa **ejs_model_identescalon24**, realizar un diseño PID óptimo de la misma. La herramienta permite realizar un ajuste manual del PID por si es necesario realizarlo más robusto, ya que tiene unos valores de margen de ganancia y margen de fase preestablecidos.

- **PID_Arduino**: La función de esta aplicación consiste en obtener los valores de las lecturas analógicas o digitales de los sensores y actuadores en pantalla y realizar grabaciones de ellas para obtener así los datos ante un ensayo de escalón en formato .txt. Para poder comunicarse con el Arduino, se debe realizar un programa mediante a la aplicación de software del Arduino. Además, esta aplicación Este cogido se encuentra en el Anexo.

Entonces, para poder realizar el ensayo de escalón, se enciende la instalación, y se ajusta el nivel del depósito para que este sea constante en relación del agua que entra y la que sale. Una vez ajustado, se aumentará el caudal con un escalón de tensión de la bomba mediante un cambio

en la referencia, hasta que la lectura del caudalímetro sea constante y una vez lo sea, se vuelve a la posición inicial, apareciendo en la gráfica dos escalones debido al aumento y disminución del caudal circulante. Con los datos obtenidos y la ayuda del programa **ejs_model_identescalon24**, se obtiene la curva de transferencia del mismo.

Estas son las distintas pruebas llevadas a cabo para poder comprobar su correcto funcionamiento:

En primer lugar, se realizarán una serie de ensayos escalón variando el periodo de muestreo para poder observar la obtención de datos por parte del caudalímetro.

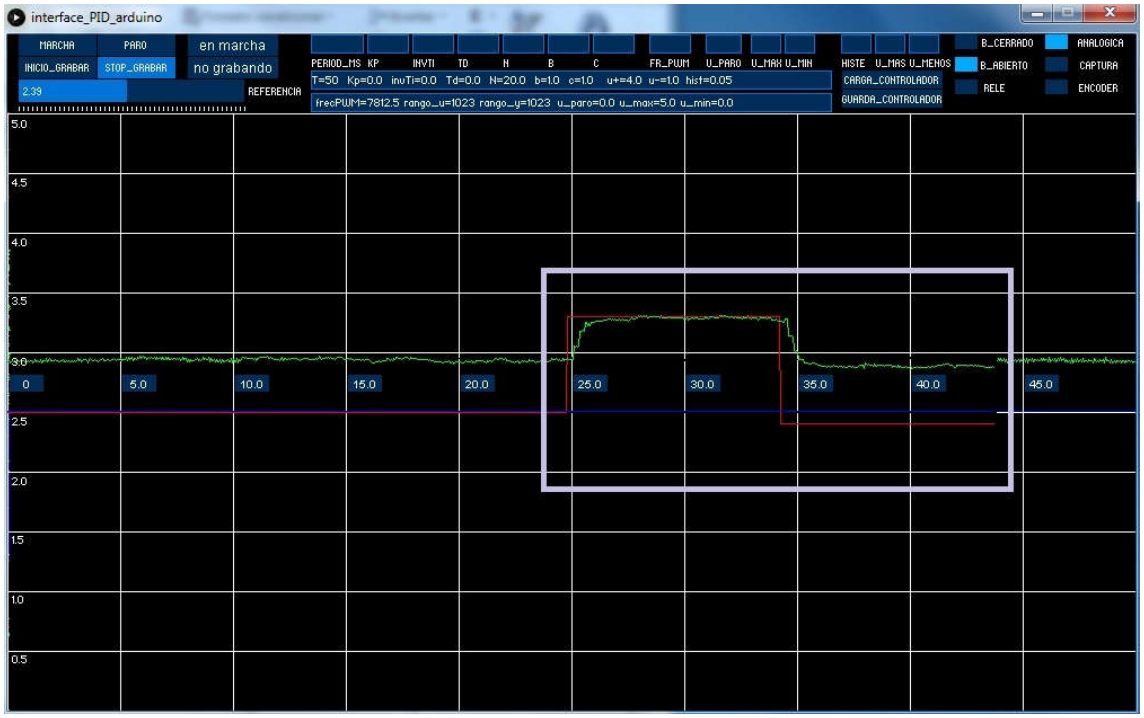



*Figura 10. DATOS 50ms B.A. B12V + CAU*

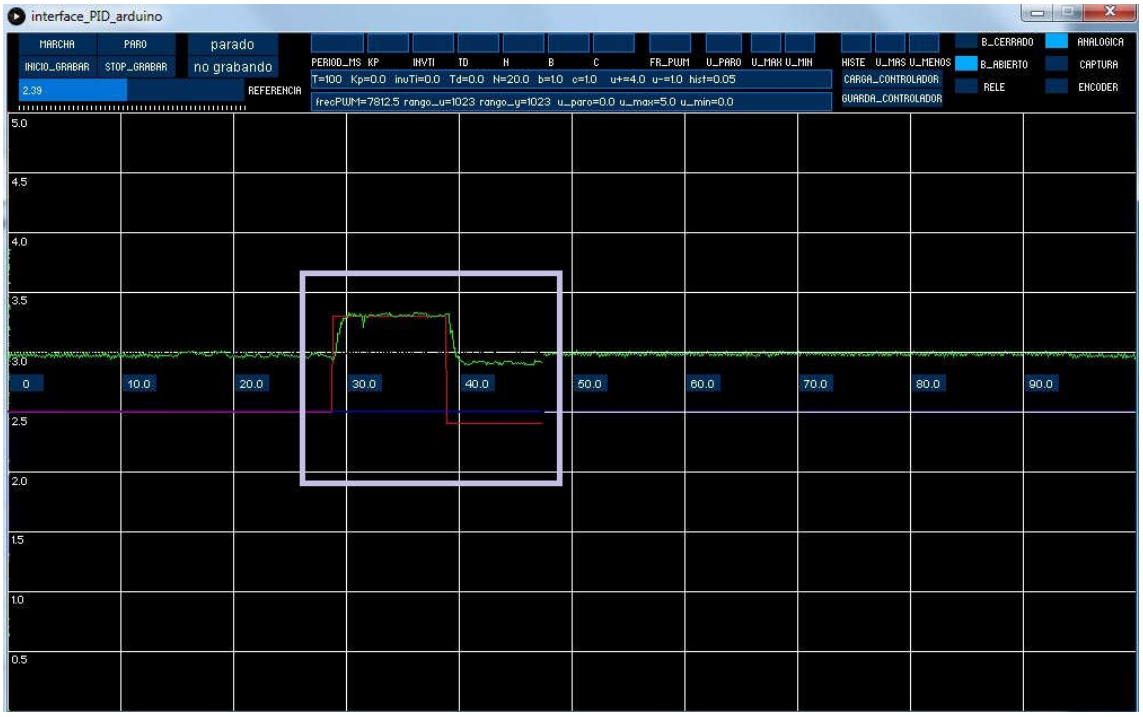



*Figura 11. DATOS 100ms B.A. B12V + CAU*

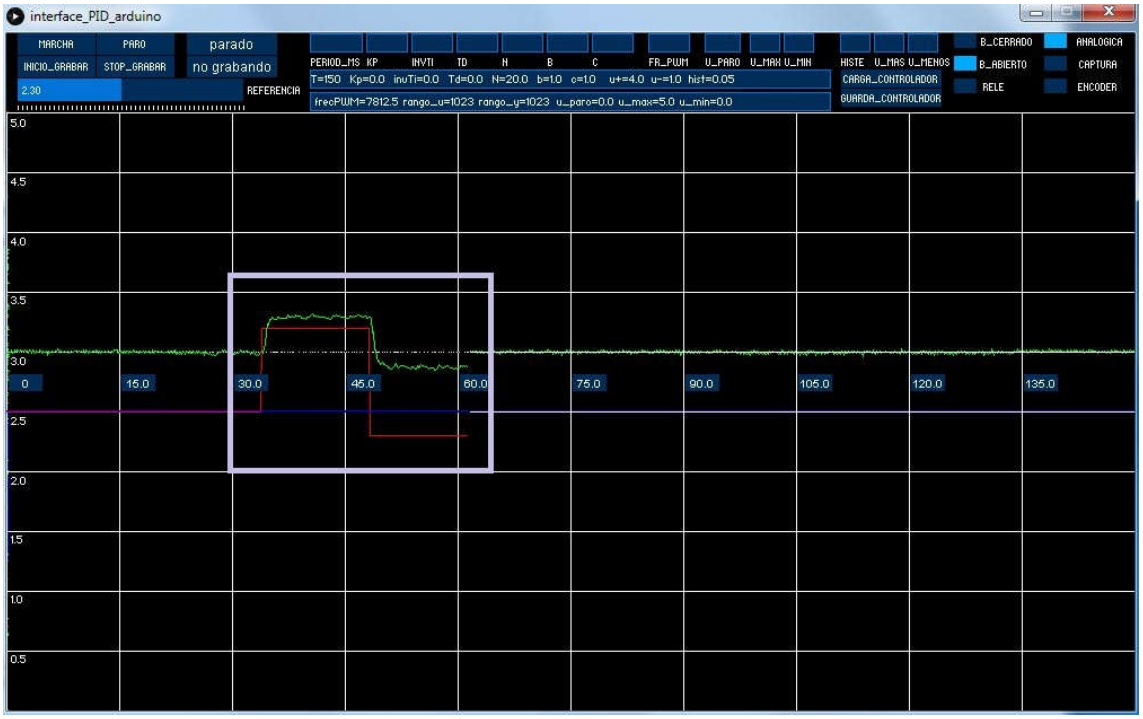



*Figura 12. DATOS 150ms B.A. B12V + CAU*

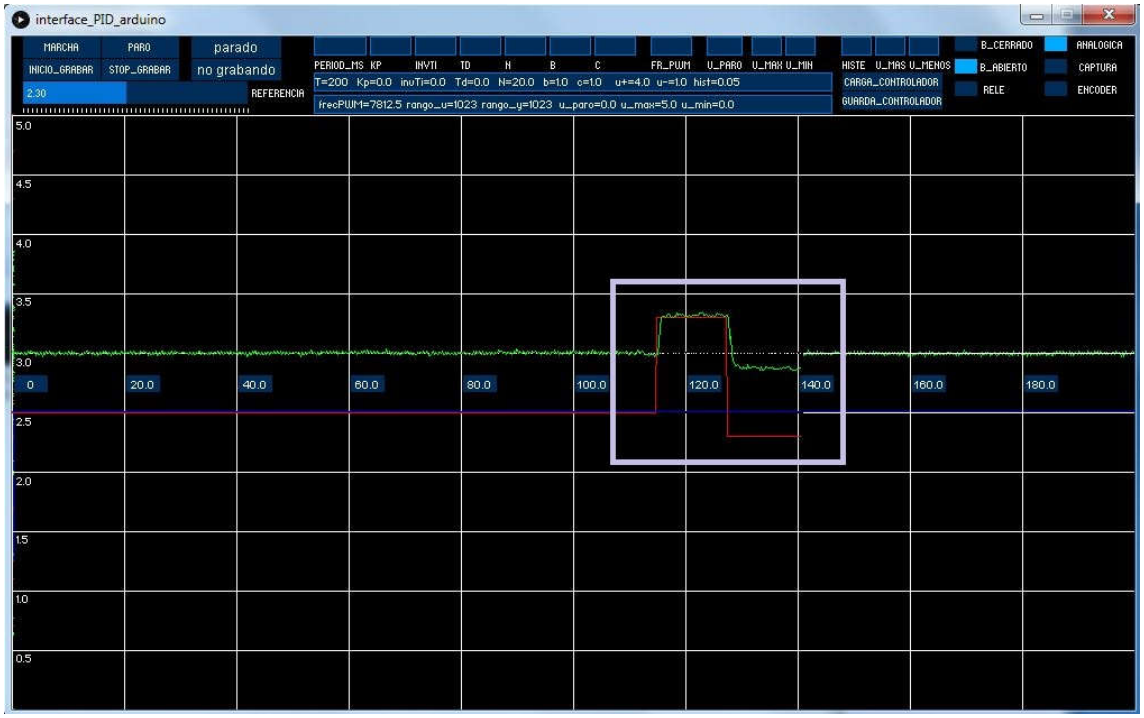



*Figura 13. DATOS 200ms B.A. B12V + CA*

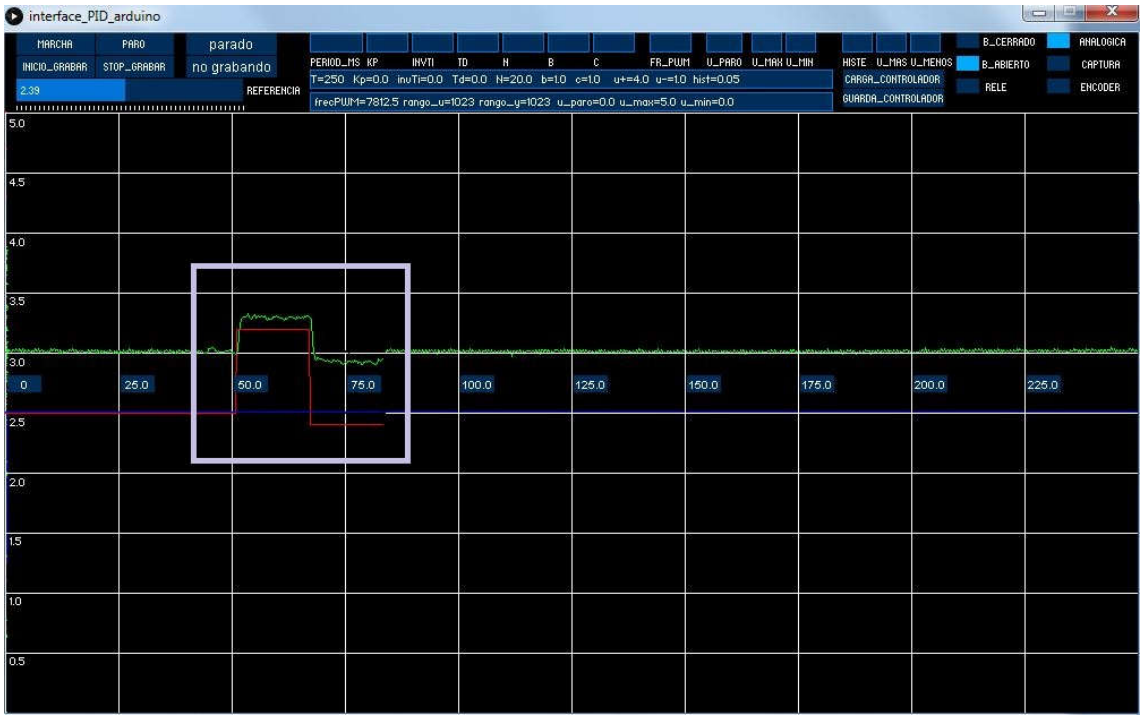



*Figura 14. DATOS 250ms B.A. B12V + CAU*

Estas imágenes representan los datos obtenidos por la aplicación **PID_Arduino**, todos ellos realizados en bucle abierto, con los que gracias a ellos obtendremos a continuación la curva

de transferencia de los mismos. Como se puede apreciar en la gráfica, los datos de color verde, son la lectura que está obteniendo el caudalímetro y la roja es la referencia, que en este caso es la tensión que llega a la bomba. A continuación, se muestras los datos obtenidos por la aplicación gracias a la herramienta Java **ejs_model_PIDFREC2053**.

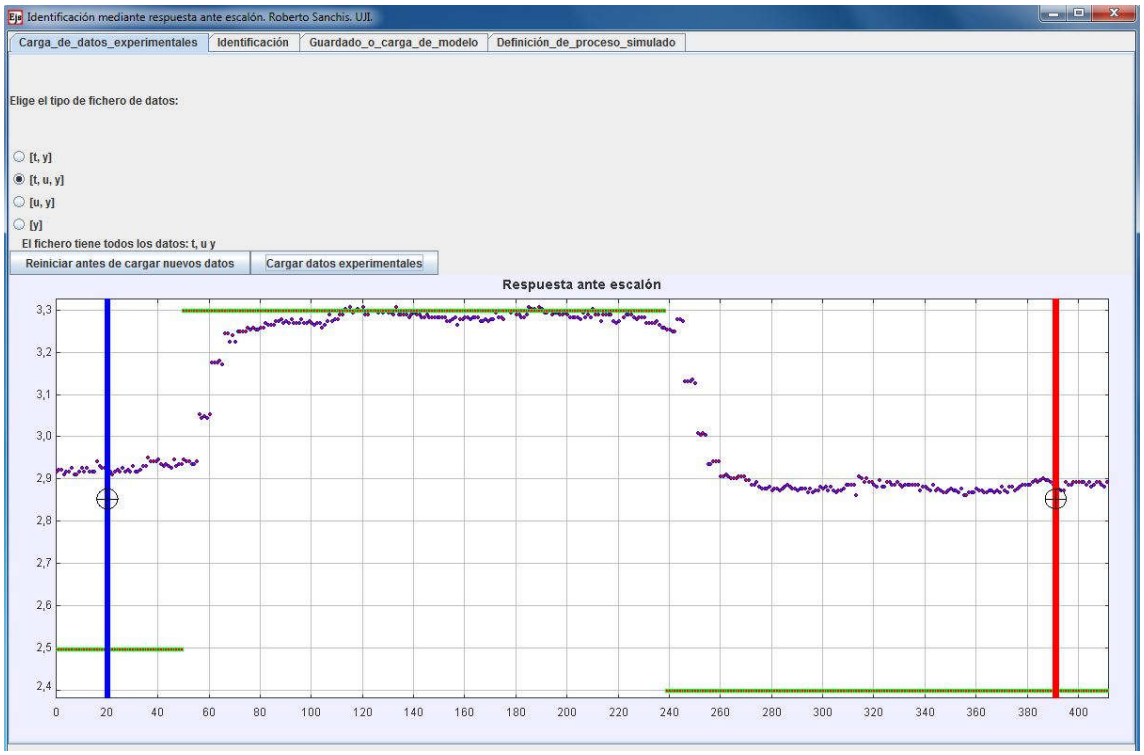



*Figura 15. DATOS ESCALON 50ms B.A. B12V + CAU*

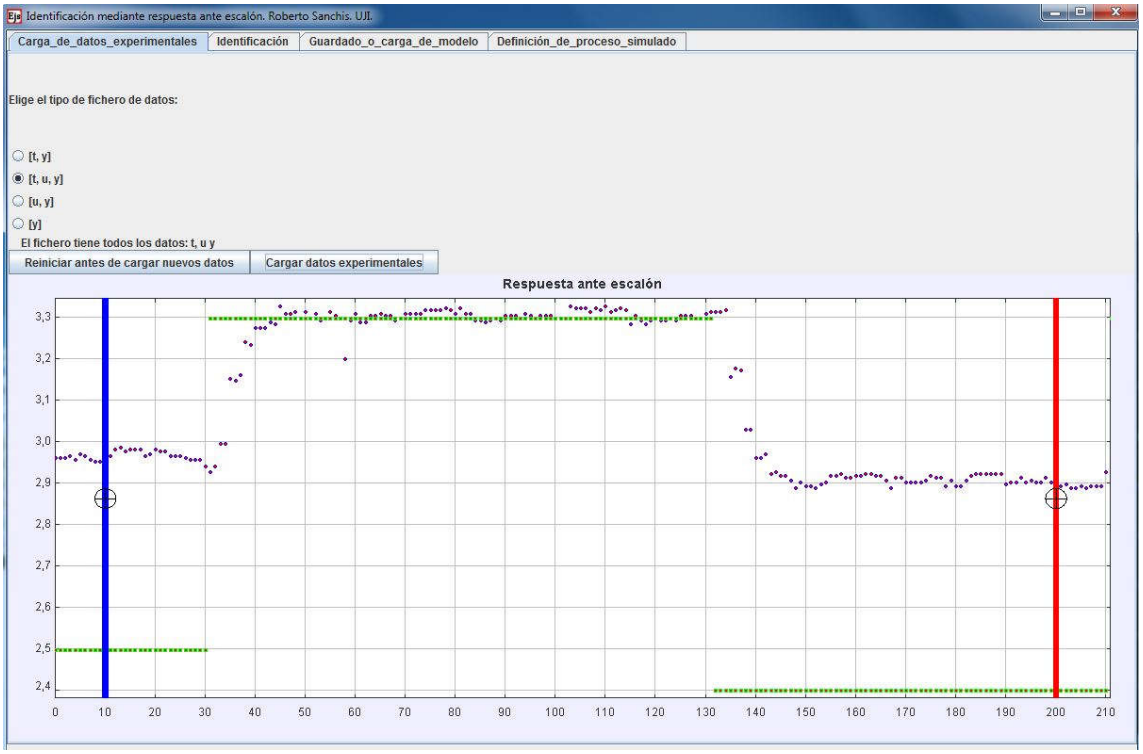



*Figura 16. DATOS ESCALON 100ms B.A. B12V + CAU*

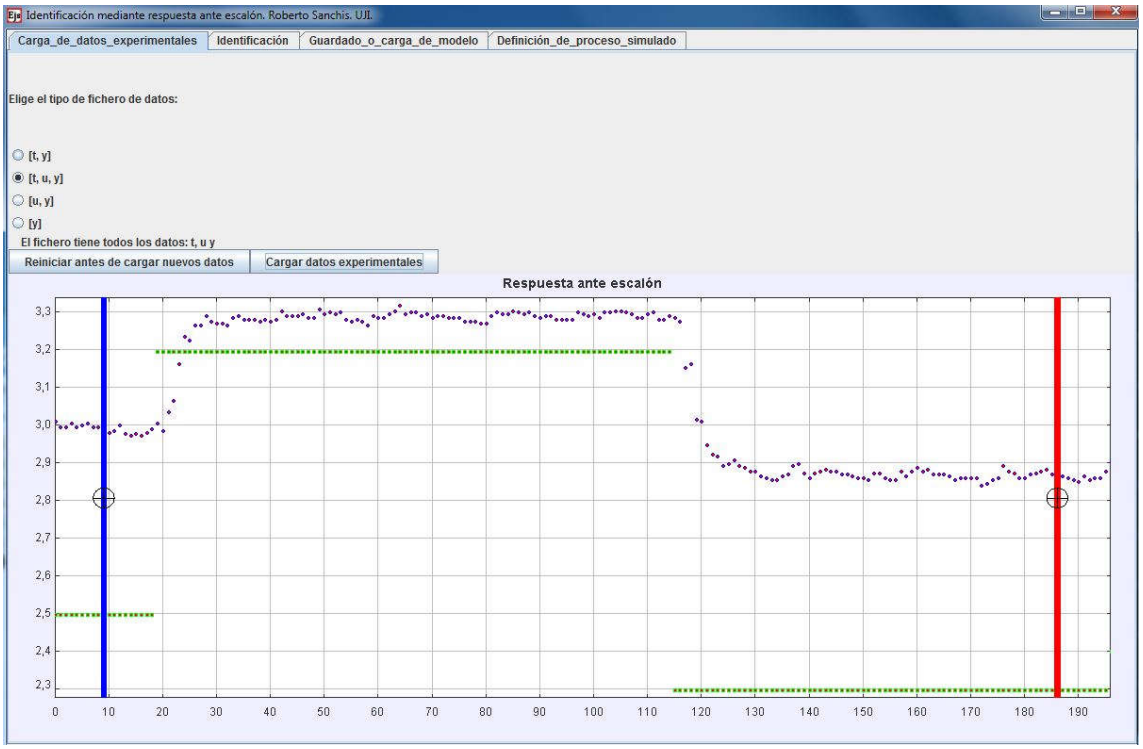



*Figura 17. DATOS ESCALON 150ms B.A. B12V + CAU*

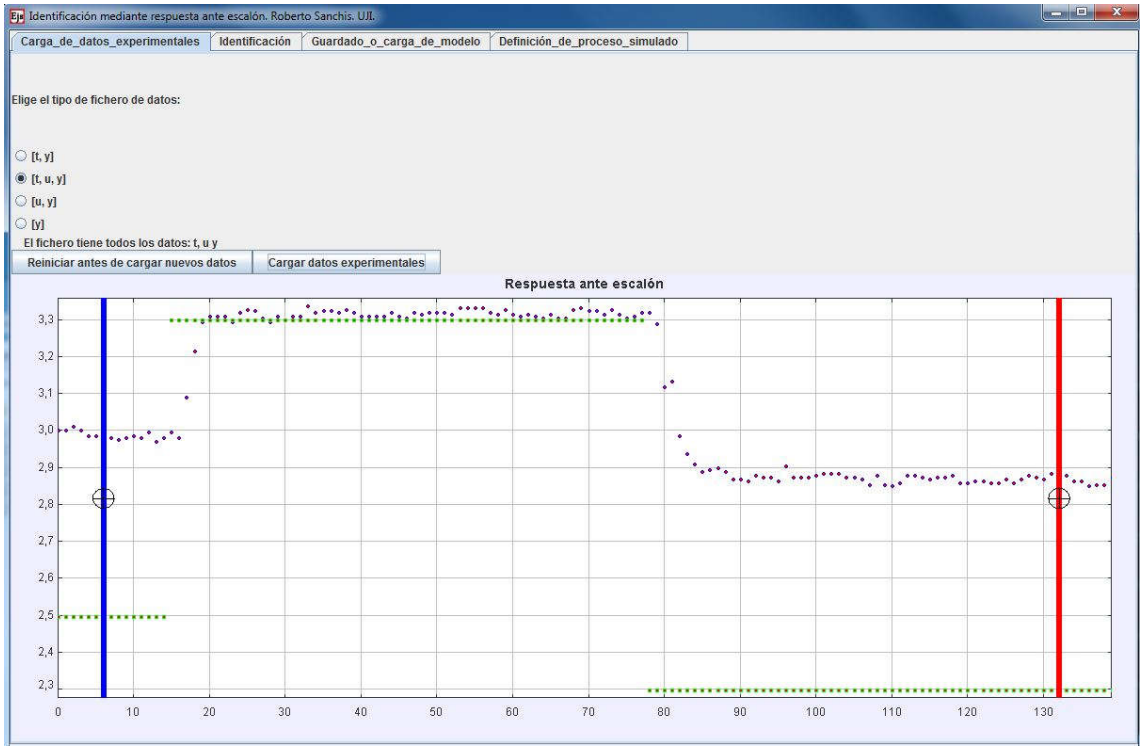



*Figura 18. DATOS ESCALON 200ms B.A. B12V + CAU*

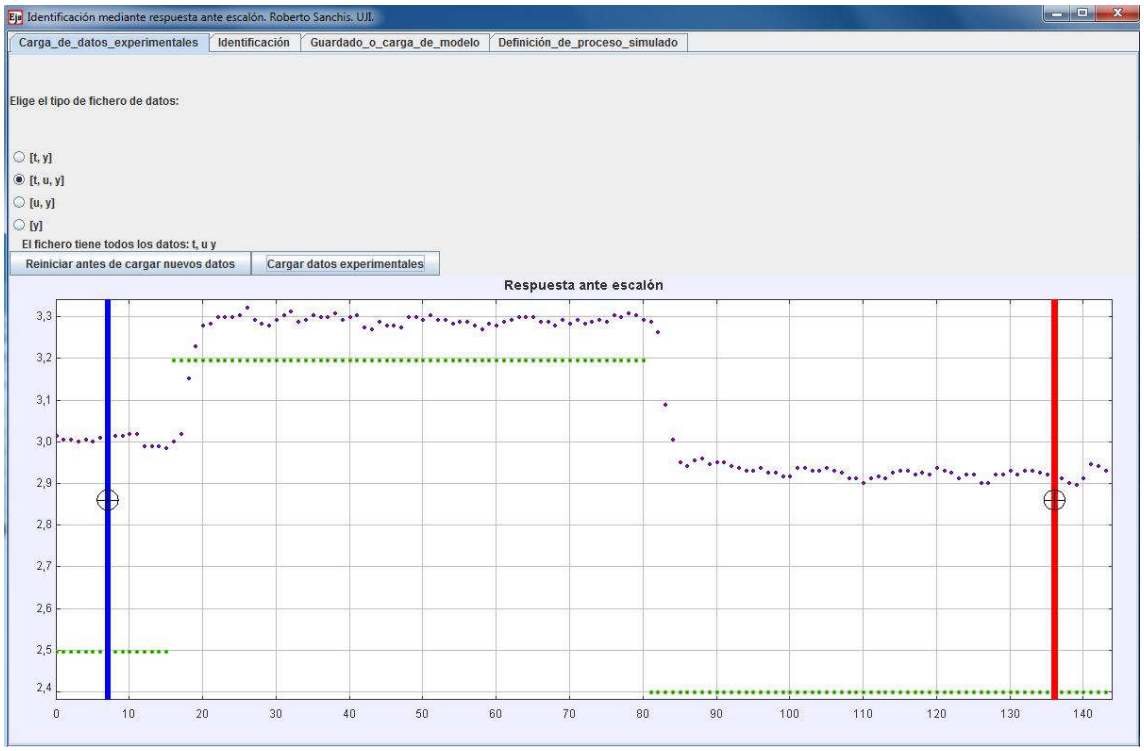



*Figura 19. DATOS ESCALON 250ms B.A. B12V + CAU*

Como se puede apreciar en las siguientes gráficas, en las primeras comprobaciones en las que se obtienen los datos con 50, 100, 150 y 200 ms de periodo de respuesta, los datos obtenidos por el caudalímetro se juntan por lo que los datos obtenidos acerca de las medidas no son correctos y no forman bien la curva de transferencia, esto se debe a que el caudalímetro no puede llegar a trabajar con un periodo de muestreo menor y por eso no se pueden realizar las pruebas con un periodo de respuesta tan bajo. Para la obtención de datos, se realizarán con un tiempo de muestreo de 250ms. Una vez aquí, se dispone a obtener la curva de transferencia obtenida en el ensayo de escalón de 250ms. Para llevarla a cabo se utilizará la aplicación anterior para ajustar esa curva de transferencia. El resultado obtenido es el siguiente:

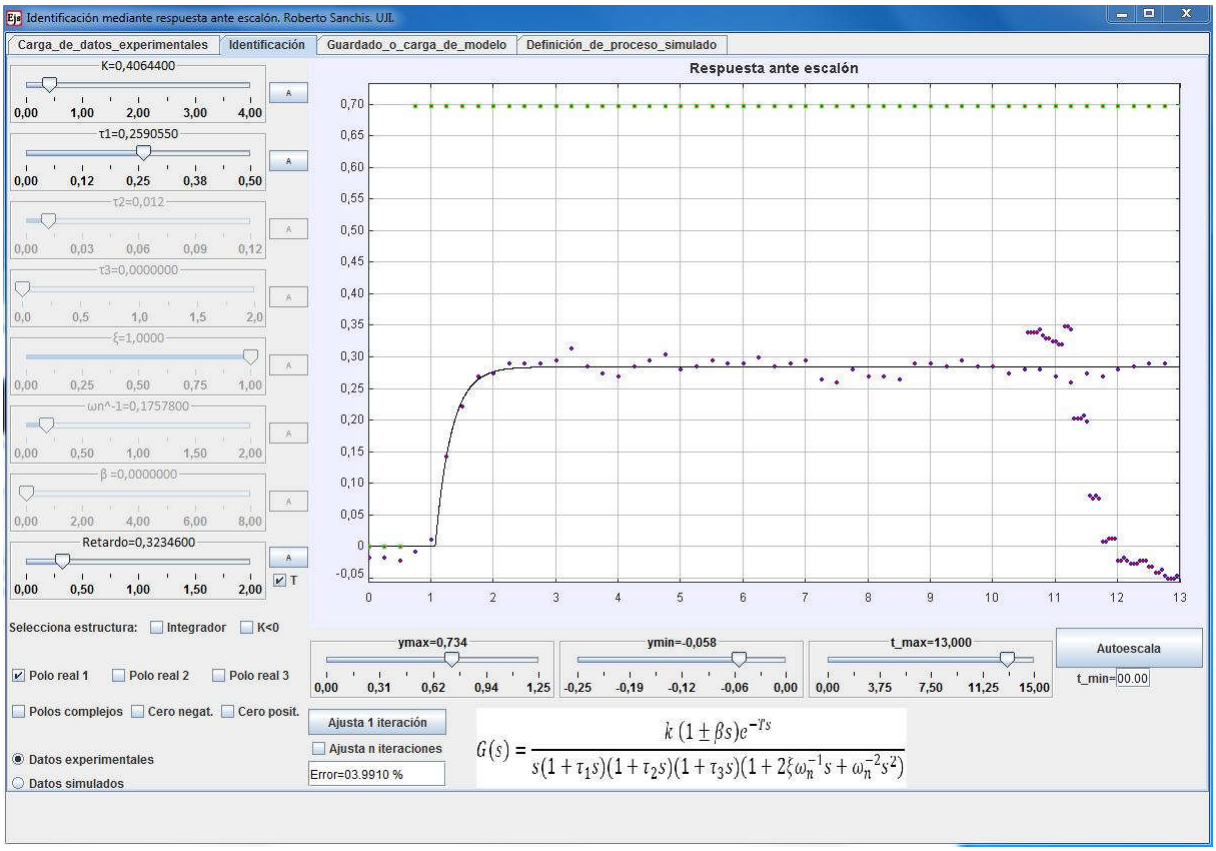



*Figura 20. RESPUESTA ANTE ESCALON 250ms B.A. B12V + CAU*

Donde la curva de transferencia obtenida es la siguiente:

$$G(s) = \frac{0.4064}{1 + 0.259s} * e^{-0.323s}$$

Una vez obtenida la función de transferencia, se introduce en el programa, **ejs_model_PIDFREC2053**. Con la ayuda del mismo, se diseña el PID para que la desviación o el error entre la acción de control y la referencia sea el mínimo posible. Lo que ocurre es que el programa realiza un ajuste optimo del PID, que, cuando se prueba aparecen unas sobre

oscilaciones muy elevadas ante un cambio de referencia. A continuación, se muestra el diseño del PID por parte de la aplicación y cómo se comporta ante cambios en la referencia.

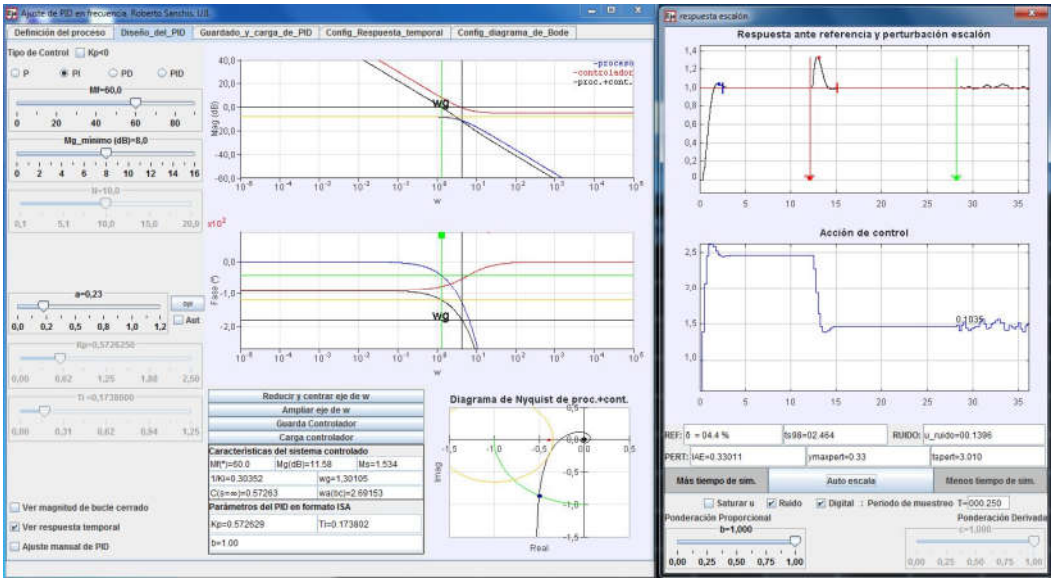



*Figura 21. Ajuste PID óptimo*

Los resultados para el PI resultante son: Kp = 0.572629, Ti = 0.173802, b = 1, a = 0.23, N = 10, Mf = 60 y Mg = 8.

En el caso de los ensayos en bucle cerrado, la acción de control pasa a ser la gráfica verde, el caudal que está circulando por el caudalimetro, la referencia, la gráfica azul y la gráfica roja, es la tensión utilizada por la bomba.

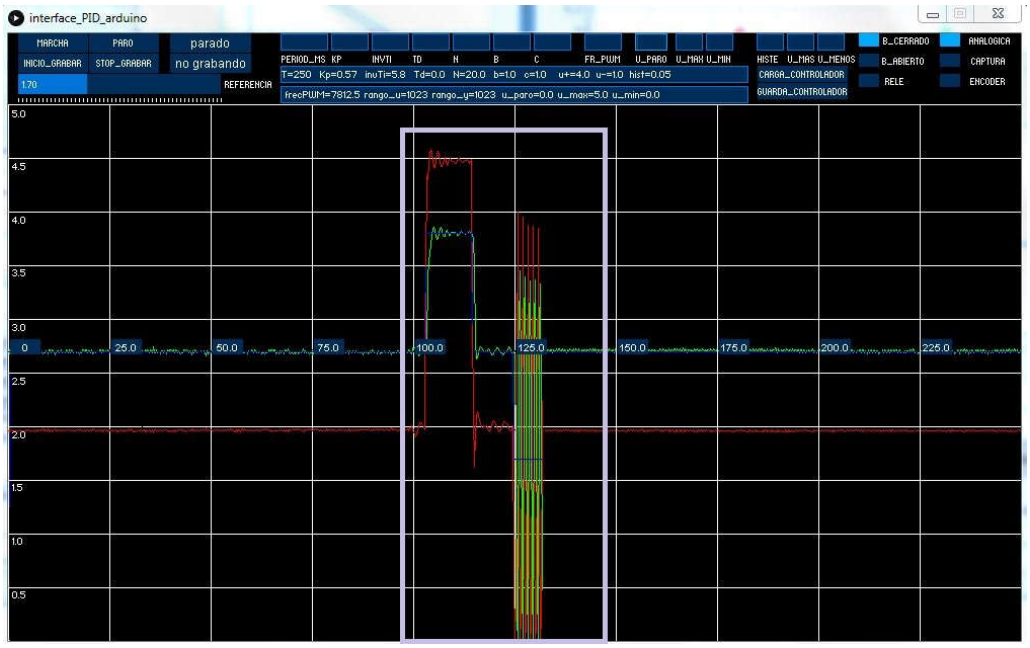



*Figura 22. DATOS B.C. B12V + CAU con sobre oscilaciones*

A la hora de realizar las pruebas en bucle cerrado, aparecen estas sobreoscilaciones ante un cambio de referencia de bajo nivel. Por lo que se decide realizar otro diseño PID más robusto.

El ajuste del PID se hará de forma manual hasta que no aparezcan sobreoscilaciones ni desviaciones y que el error sea el mínimo posible. Como las sobreoscilaciones siguen apareciendo, se decide diseñar un PID más robusto aumentando el margen de fase de 60 a 80. De esta manera obtenemos un diseño de PID que funciona correctamente, pero hace que la respuesta sea un poco más lenta.

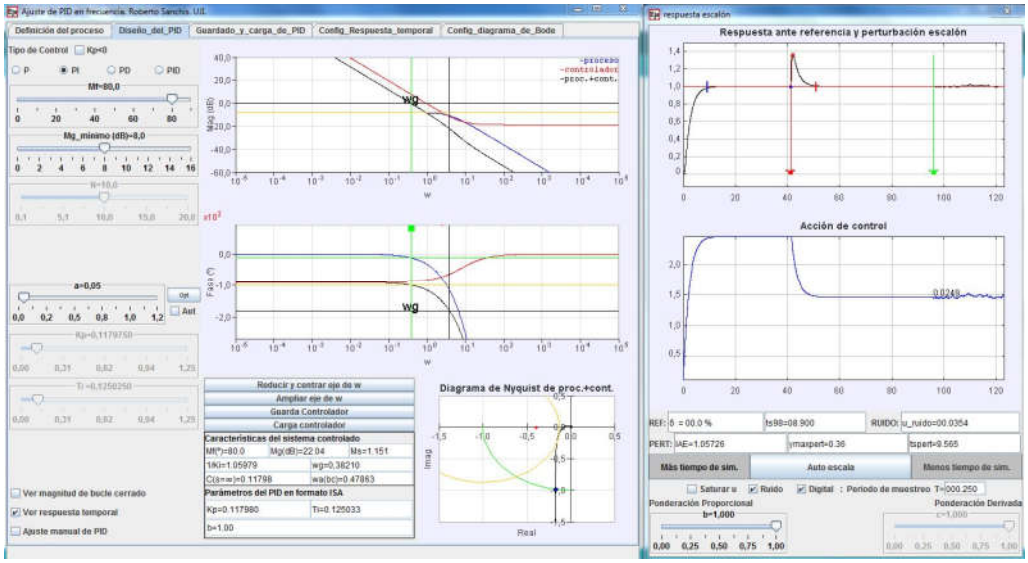



*Figura 23. Ajuste PID manual con mayor robustez*

Los resultados para el PI resultante son: Kp = 1.05979, Ti = 0.125033, b = 1, a = 0.05, N = 10, Mf = 80 y Mg = 8.

El resultado obtenido a continuación por el ensayo en un bucle cerrado es el siguiente:

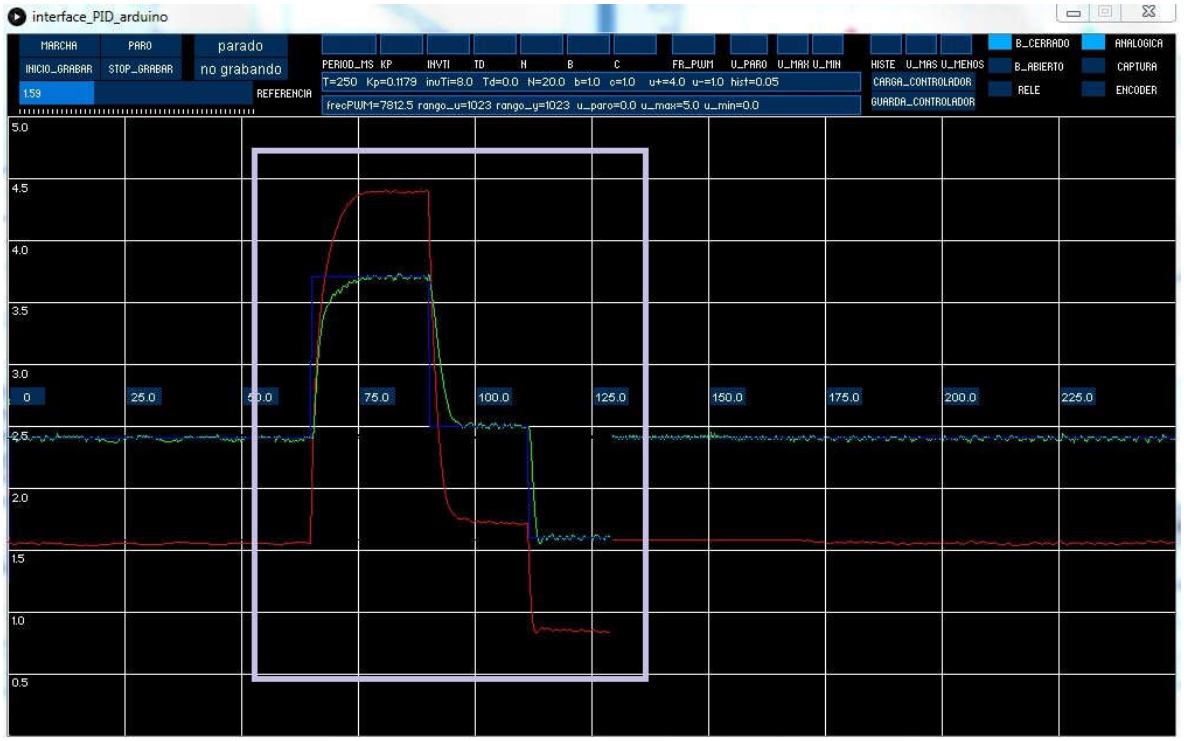



*Figura 24. DATOS B.C. B12V + CAU sin sobre oscilaciones*

Como se puede apreciar ahora en las gráficas, las sobreoscilaciones han desaparecido y la acción de control se ajusta perfectamente a la referencia.

Al finalizar el ajuste del PID, se deberá modificar el código del Arduino para que el caudalímetro y el sensor de nivel funcionen en cascada, por lo que se añadirán unas líneas de código nuevas y se colocará el PID obtenido por caudalímetro de forma manual. Los cambios realizados en el código del Arduino, se encontrarán en los Anexos

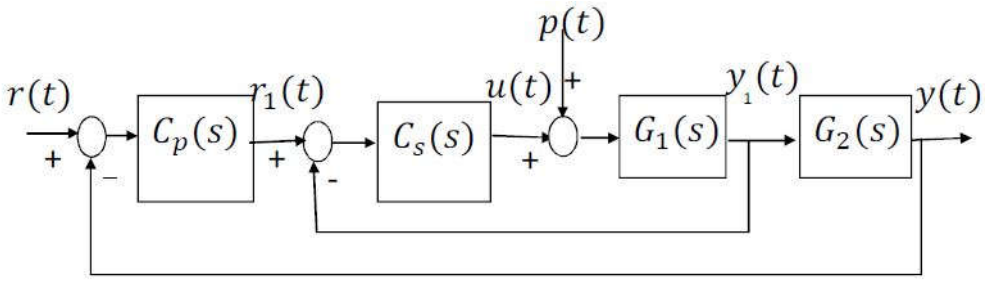



*Figura 25. Configuración PID en cascada*

### 1.6.3.3. Comprobaciones y bucle de control en el sensor de nivel capacitivo

Una vez realizados los cambios en el programa, se realizan las mismas pruebas anteriores, pero con el sensor de nivel (El que ahora se encuentra en el laboratorio).

Para empezar a realizar las pruebas con el sensor de nivel, es necesario ajustar primero la entrada y salida de caudal en instalación de manera que la lectura del sensor de nivel sea constante. A partir de ahí, el primer paso consiste es realizar el ensayo de escalón. Los datos obtenidos por la aplicación **PID_Arduino**, son los siguientes:

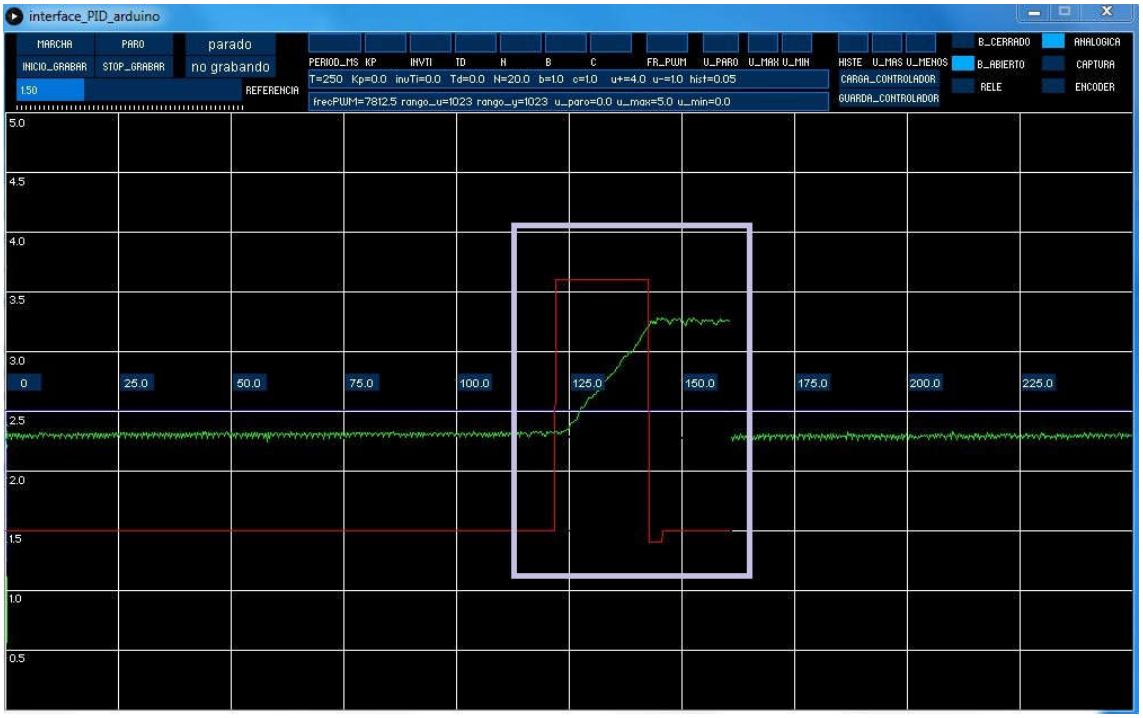



*Figura 26. DATOS B.A. B12V + CAU + SENSOR NIVEL*

En el caso de la obtención de datos en bucle abierto, una vez realizados los cambios al código del Arduino, la gráfica de color rojo, pertenece a la referencia, que es la tensión con la que está trabajando la bomba y la gráfica verde, representa el sensor de nivel, indicando en cada momento el nivel de agua que se encuentra en el depósito.

Realizando las comprobaciones, se puede observar que el sensor de nivel no marca correctamente el nivel de agua que existe en el depósito, ya que este una vez detecta al nivel, le cuesta mucho tiempo mostrarlo en pantalla e incluso una vez se llega a un nivel deseado, este continúa realizando las lecturas, superando ese nivel.

Una vez obtenidos los datos, el siguiente paso consiste en obtener la curva de transferencia a partir de la aplicación **ejs_model_identescalon24**. El resultado obtenido es el siguiente:

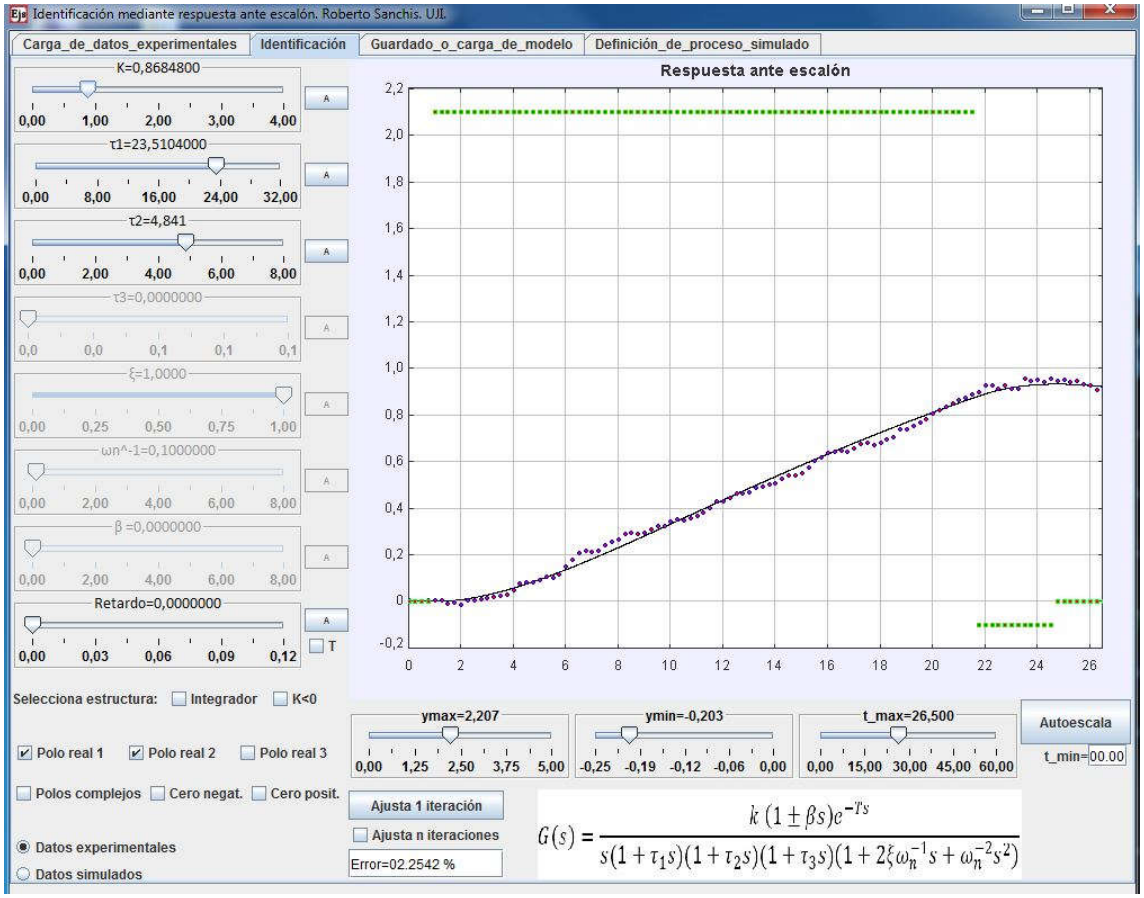



*Figura 27. RESPUESTA ANTE ESCALON 250ms B.A. B12V + CAU + SENSOR*

Tras realizar el ajuste, la curva de transferencia obtenida es la siguiente:

$$G(s) = \frac{0.8684}{1 + 28.351s + 113.8038s^2}$$

Una vez obtenida la curva de transferencia con la ayuda de la aplicación **ejs_model_PIDFREC2053**, se puede realizar un diseño PID para que la desviación o el error entra la acción de control y la referencia sea mínimo. La solución adoptada en el siguiente caso, es:

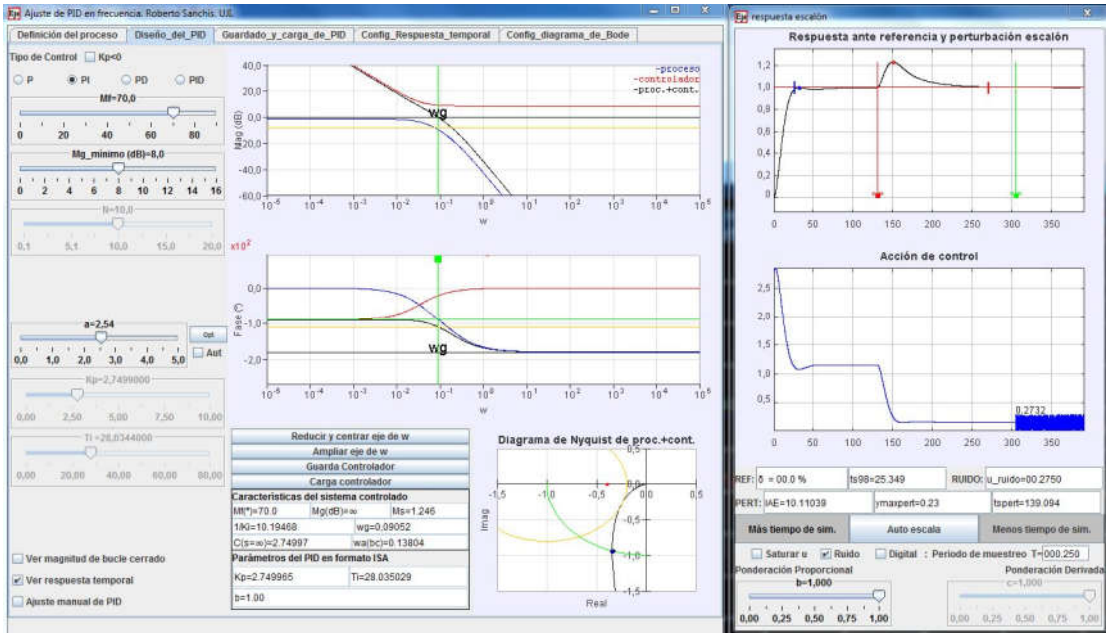



*Figura 28. Ajuste PID B12V + CAU + SENSOR*

Los resultados para el PI resultante son: Kp = 2.7499, Ti = 28.035, b = 1, a = 0.05, N =10, Mf = 70 y Mg = 8.

Y una vez obtenido el PID resultante, se añaden los nuevos valores en la aplicación **PID_Arduino** para poder comprobar el correcto funcionamiento del mismo.

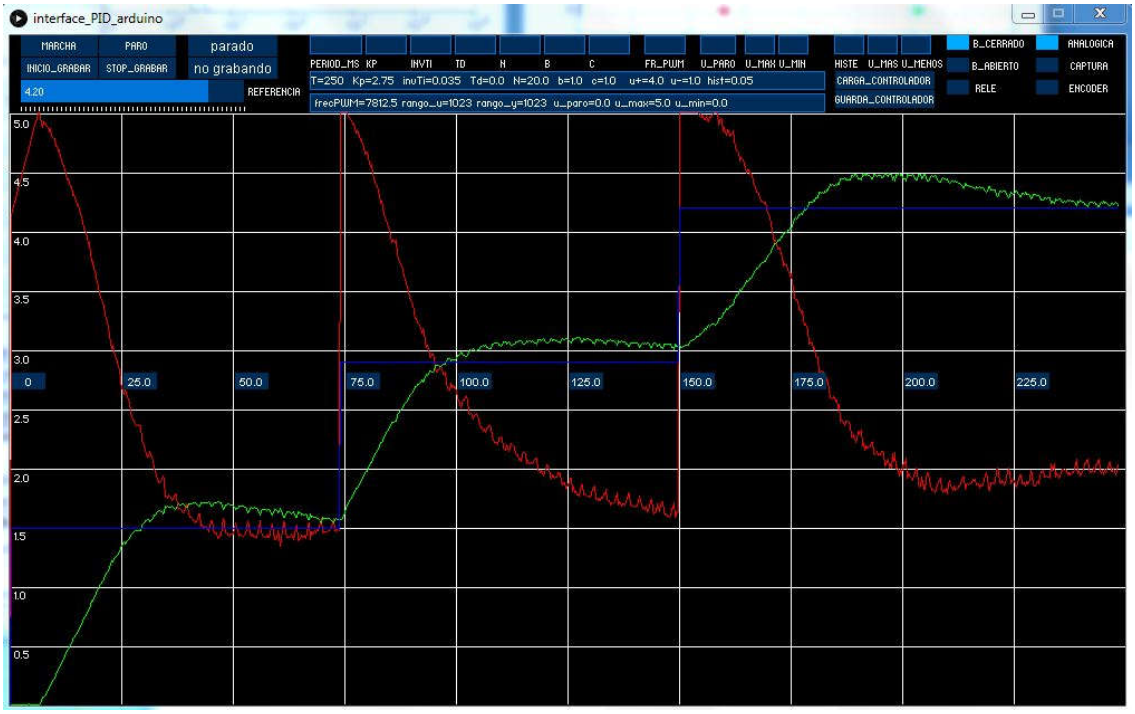



*Figura 29. DATOS PID B12V + CAU + SENSOR*

Las gráficas, cuando realizamos el ensayo en bucle cerrado varían, en este caso la gráfica de color verde corresponde al sensor de nivel, la azul a la referencia y la roja a la tensión con la que está trabajando la bomba.

Tras realizar las pruebas, se pueden obtener unos resultados en los que se puede apreciar que el sensor de nivel tiene una respuesta muy lenta y que cuando este llega a la altura de la referencia, lo sobrepasa durante un tiempo y posteriormente se ajusta al nivel de referencia, por lo que se decide obtener el nuevo sensor de nivel capacitivo e intentar resolver este problema.

### 1.6.4. Obtención de nuevos componentes, comprobaciones y nuevos bucles de control

Una vez terminadas las comprobaciones anteriores, se decide obtener los nuevos componentes para el diseño del proyecto y realizar todos los cambios oportunos a la estructura. Unos de los cambios que se va a realizar en el proyecto, es la utilización de dos depósitos, por lo que, para poder bombear el agua a los distintos depósitos, una de las soluciones consiste en conectar un T en serie con la bomba y a continuación dos electroválvulas todo/nada o la utilización de dos bombas distintas en cada depósito para así bombear el agua. La función de estas, consiste en bombear el agua en un solo depósito en concreto, y cuando este esté lleno, empezar a llenar el otro. Los cambios realizados en la parte de la estructura, son bastante sencillos. En primer lugar, sustituir el antiguo sensor de nivel por el nuevo. Como ambos vienen con la misma unión roscada, basta con sustituirlo por el otro. A continuación, se coloca una nueva bomba en serie con otro caudalímetro para poder controlar el caudal de salida de la instalación.

Para poder colocar las electroválvulas, se han de realizar algunos cambios en la estructura. Los cambios estructurales consisten en realizar las distintas unidos mediante tubos termoestables del mismo diámetro que los usado anteriormente y su sujeción mediante bridas.

Los nuevos componentes son:

- Bombas sumergibles de 24V
- Caudalímetro de ultrasonidos
- Sensor de nivel capacitivo
- Electroválvulas todo/nada de riego

Una vez obtenidos los nuevos componentes, antes de realizar el montaje final, se testean los componentes. Para ello se realizan las pruebas utilizadas en las primeras comprobaciones.

### 1.6.4.1. Comprobaciones en las bombas de 24V

A la hora de probar la bomba, se vuelven a realizar las conexiones necesarias para el correcto funcionamiento de la misma, y como la bomba obtenida, también es muy económica, esta no trae consigo una curva de caudal. Por lo que al igual que en la anterior se ha de realizar una curva de caudal manualmente. Así que, con la ayuda de un multímetro y el caudalímetro, se realiza la curva de caudal.

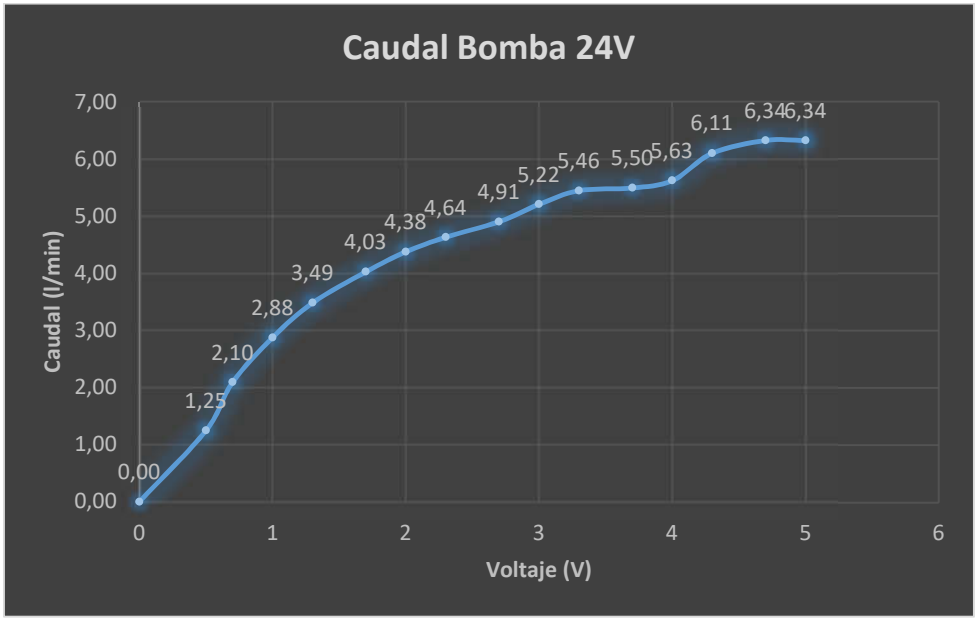



Tabla 2. Curva caudal bomba 24V

### 1.6.4.2. Comprobaciones y nuevo bucle de control en el caudalímetro

Al obtener la curva de caudal, el siguiente paso consiste en diseñar un nuevo PID para la nueva bomba junto con el caudalimetro. Para ello, con la ayuda de la aplicación y de los programas Java, se procede a su diseño. En primer lugar, se realiza un ensayo de escalón. Con los programas Java, se obtiene la nueva curva de transferencia y posteriormente el diseño del PID óptimo para que la desviación o el error entre la acción de control y la referencia, sea el mínimo posible.

Como ya sabemos con anterioridad, el caudalímetro que vamos a utilizar, una vez se intentan obtener los datos con un tiempo de respuesta muy pequeño, no realiza bien las medidas pertinentes, por lo que ahora, el periodo de muestreo que vamos a utilizar, será de 250ms.

Una vez comentado el apartado anterior, se dispone a realizar el ensayo de escalón con la nueva bomba junto con el caudalimetro. El resultado es el siguiente:


*Figura 30. DATOS 250ms B.A. B24V + CAU*

Una vez obtenidos los datos, se obtiene la curva de transferencia del mismo a partir de la aplicación Java **ejs_model_identescalon24**.

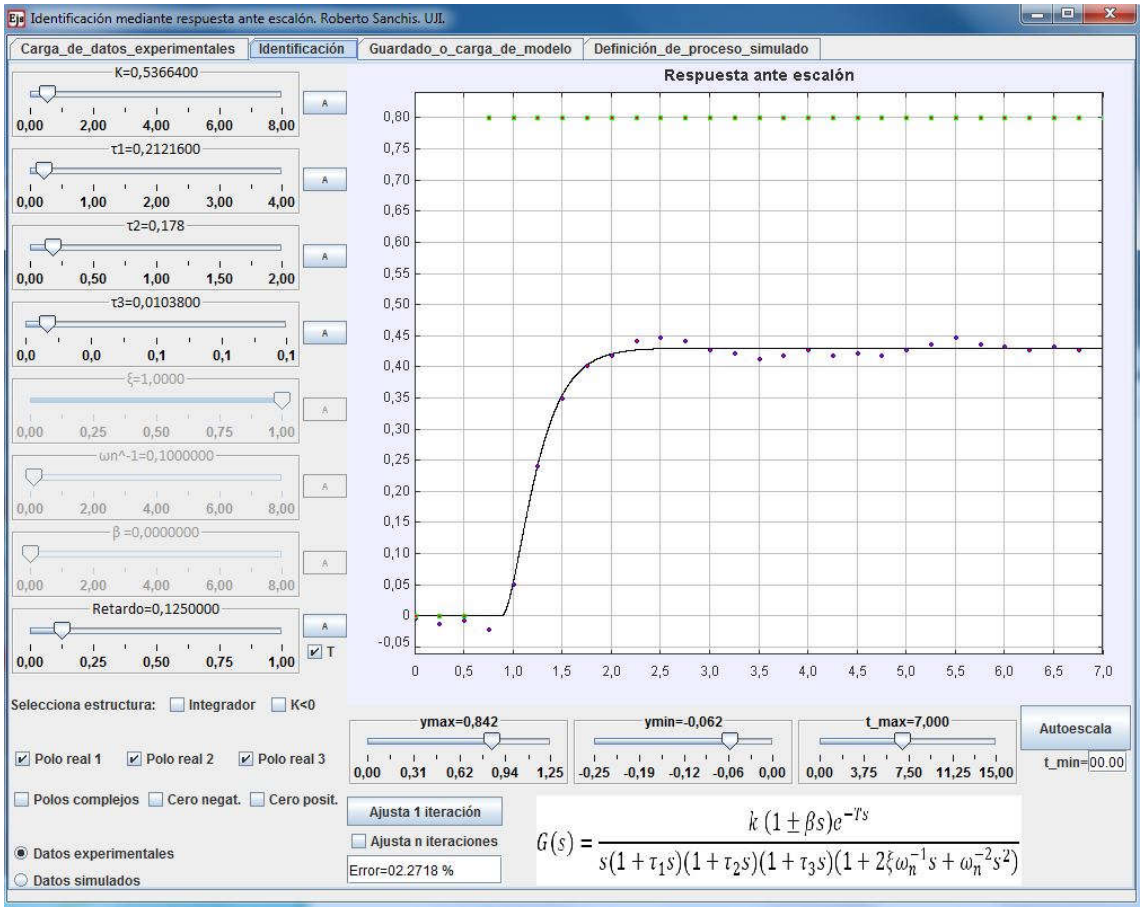



*Figura 31. RESPUESTA ANTE ESCALON B.A. B24V + CAU*

Con la que obtenemos la siguiente función de transferencia:

$$G(s) = \frac{0.5367}{1 + 0.4005s + 0.0418s^2 + 0.0004s^3} * e^{-0.125s}$$

Con curva de transferencia, el siguiente paso es diseñar el PID. Por lo que con la ayuda de la aplicación Java **ejs_model_PIDFREC2053**, se introducen los datos de la función de transferencia obtenida y se ajusta el PID. El resultado obtenido es el siguiente:

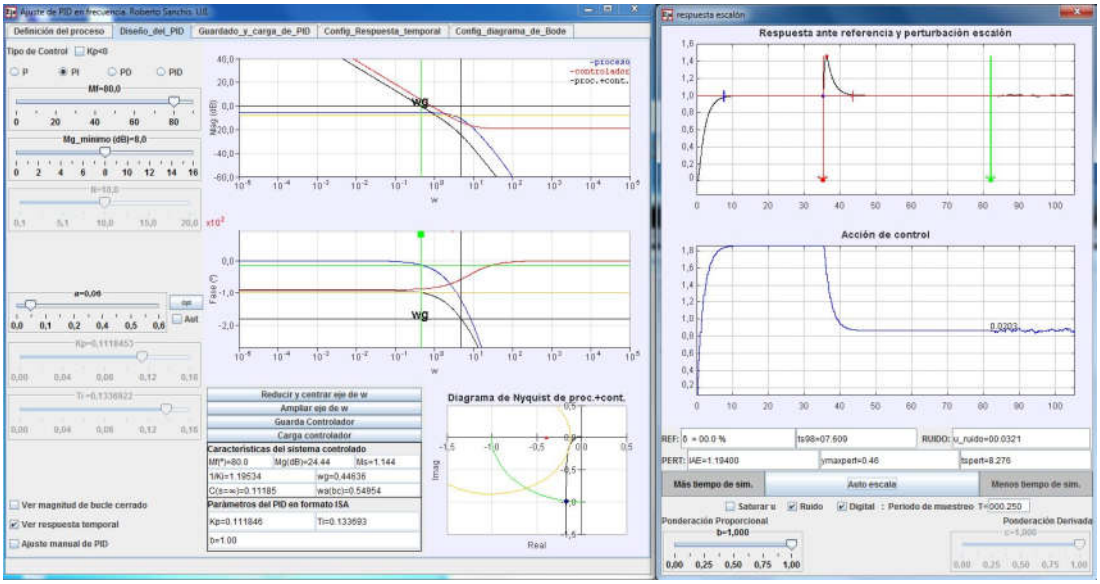



*Figura 32. Ajuste PID B24V + CAU*

Los resultados para el PID resultante son: Kp = 0.1118, Ti = 0.1337, b = 1, a = 0.05, N = 10, Mf = 80 y Mg = 8.

Con el diseño del PID necesario, se introducen los nuevos datos en la aplicación **PID_Arduino** y se comprueba su correcto funcionamiento. Como en las pruebas anteriores se tuvo que diseñar un PID robusto, en esta, directamente se utilizará un diseño parecido para evitar sobre oscilaciones en el diseño. Los datos obtenidos con la nueva configuración son los siguientes:

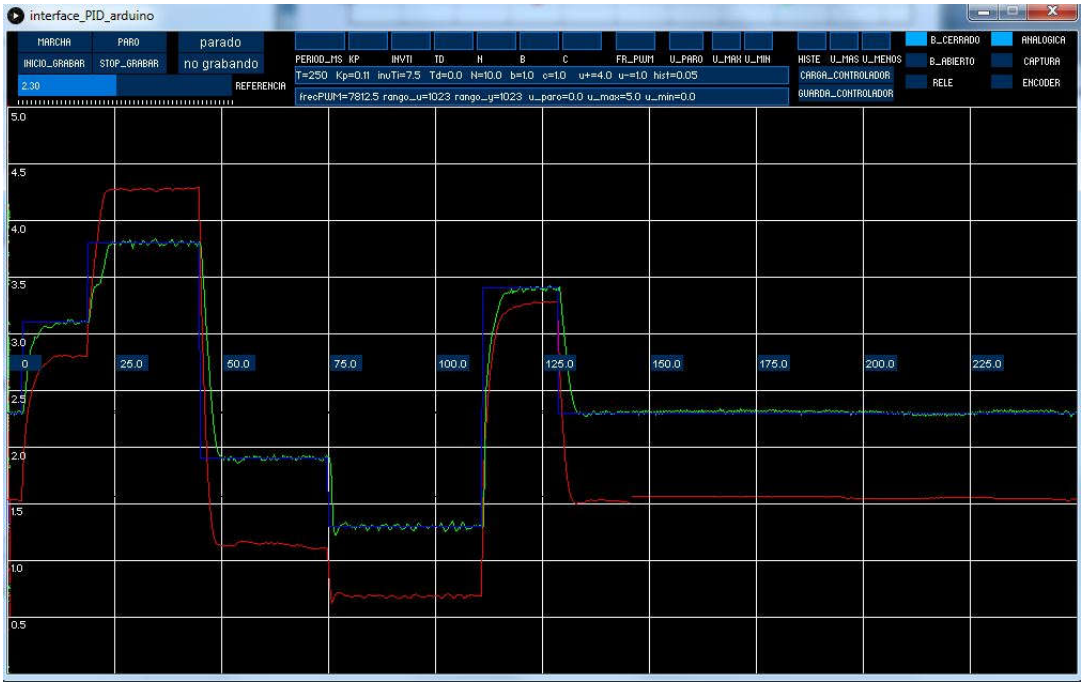



*Figura 33. DATOS PID B24V + CAU + SENSOR*

En la imagen que se muestra en pantalla, se puede observar un buen funcionamiento del diseño, en el que la acción de control se ajusta perfectamente a la referencia, por lo que el diseño de PID funciona correctamente.

Una vez finalizado el ajuste del caudalímetro junto con la bomba, el siguiente paso consiste en conectar el nuevo sensor de nivel, realizar las conexiones pertinentes y calibrarlo antes de poder realizar ningún ensayo.

### 1.6.4.3. Comprobaciones y nuevo bucle de control del sensor de nivel capacitivo

Tras realizar el diseño del PID para la bomba junto con el sensor de nivel, el siguiente paso, antes de realizar el ajuste del mismo, consiste en realizar el conexionado previo del sensor de nivel, ajuste en el depósito y calibrar el sensor de nivel en cuanto al fondo de escala y asignar donde se encuentra el nivel máximo y mínimo en el depósito que se utilizará para realizar las medidas. A la hora de realizar las conexiones, se debe tener en cuenta que este sensor ofrece una salida analógica de entre 4 mA-20 mA, por lo que, para cambiar esa salida de corriente por una de tensión, se colocará una resistencia en paralelo en la salida. Por eso, con una resistencia de 250 Ohmios, se cambió a una tensión entre 1V-5V.

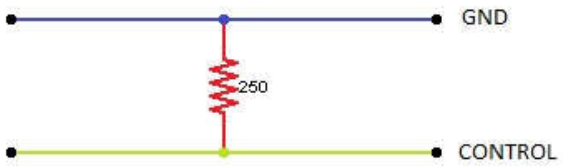



*Figura 34. De salida de corriente a salida de tensión*

Una cosa a tener en cuenta con este sensor de nivel, es que no aparecen en el manual la manera con las que están realizadas las conexiones internas, por lo que, con la ayuda del multímetro, se comprobó que las masas no estaban interconexionadas, por lo que, para no provocar un cortocircuito en el Arduino y posteriormente en el PLC (provocando la posible avería de algunos de sus componentes internos), se decide alimentar al sensor de nivel con una fuente de alimentación externa distinta. Con la ayuda del manual proporcionado por el fabricante del sensor, podemos observar que la calibración del sensor es bastante sencilla. Tan solo hay que marcarle nuestro nivel máximo y mínimo de funcionamiento y realizar una simple combinación

de botones para realizar el ajuste. Al realizar todos los pasos previos, se añaden manualmente los valores del diseño PID obtenidos en la calibración de la bomba junto con el caudalímetro. Por eso, se modifican algunos datos del código en el Arduino, los cuales se podrán encontrar en los Anexos de la memoria.

Una vez modificados los cambios, con la ayuda de los programas Java y la aplicación **PID_Arduino**, se tomarán las respectivas medidas para realizar el ajuste PID en cascada junto con el caudalimetro y la bomba. El primer paso, igual que en las pruebas anterior, consiste en realizar un ensayo de escalón con el que se obtendrá la función de transferencia del mismo. Para poder realizar el ensayo de escalón, el primer paso consiste en ajustar un nivel de agua constante del sensor de nivel, por lo que se regulará manualmente la entra y salida de caudal para que estos sean iguales. Con los cambios realizados, se obtienen los datos en bucle abierto a partir de la aplicación **PID_Arduino**.

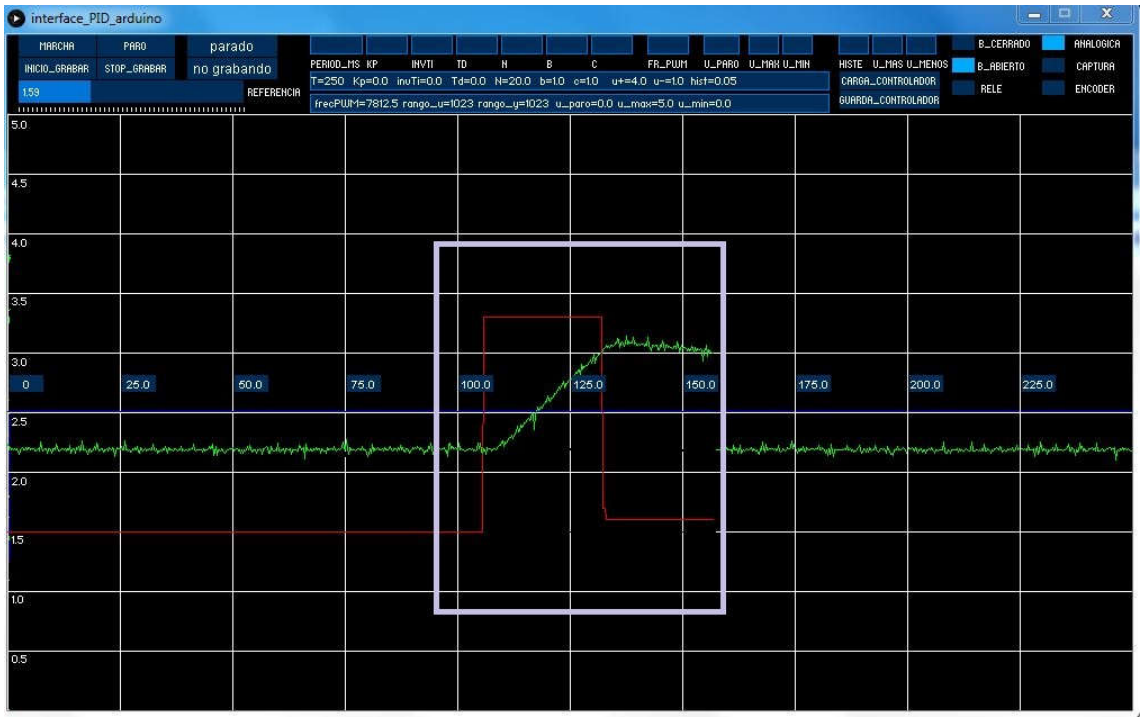



*Figura 35. DATOS 250ms B.A. B24V + CAU + SENSOR*

Una cosa que no se puede apreciar en las gráficas, es la velocidad de respuesta del sensor de nivel. En este caso cuando el agua entra en contacto con el sensor, este, inmediatamente lo reconoce y empieza a mostrar datos en pantalla. Esto no pasaba con el sensor de nivel que se ha utilizado en las pruebas, ya que esto fue una de las principales decisiones por la que se obtuvo un nuevo sensor.

En la siguiente grafica se muestran los datos obtenidos a por la aplicación **PID_Arduino**. En este caso, la gráfica de color verde, está marcando el nivel de agua que está detectando el sensor y la gráfica roja, es la referencia, que es la tensión con la que está alimentado a la bomba de agua. Una vez obtenidos los datos a partir del Arduino, se introducen dentro de la aplicación Java **ejs_model_identescalon24** y así poder realizar el ensayo de escalón**.**

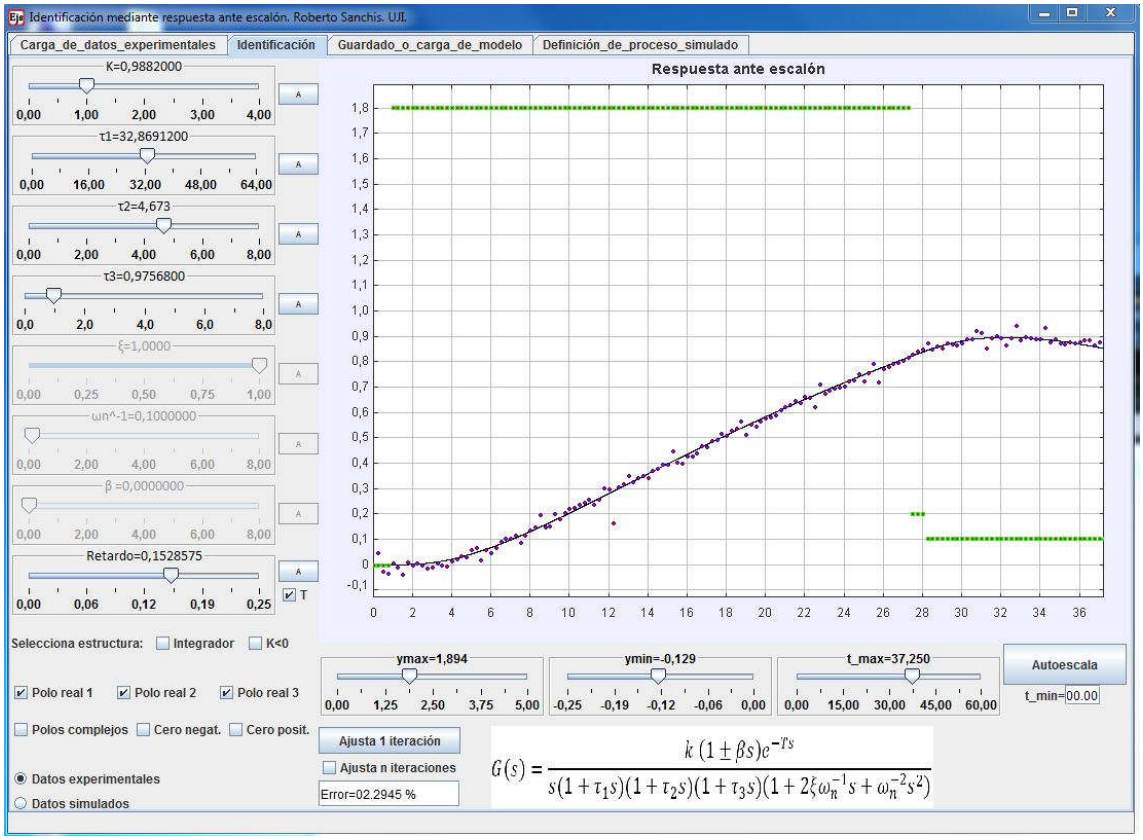



*Figura 36. RESPUESTA ANTE ESCALON B.A. B24V + CAU + SENSOR*

Con los que se obtiene la siguiente función de transferencia:

$$G(s) = \frac{0.9882}{1 + 38.5189s + 190.2463s^2 + 149.8873s^3} * e^{-0.1529s}$$

Al obtener la función de transferencia, a continuación, se realizará el diseño de PID óptimo, por lo que, con el programa **ejs_model_PIDFREC2053**, se obtendrá el diseño del PID el cual nos permite obtener una desviación mínima o error mínimo entre la acción de control y la referencia.

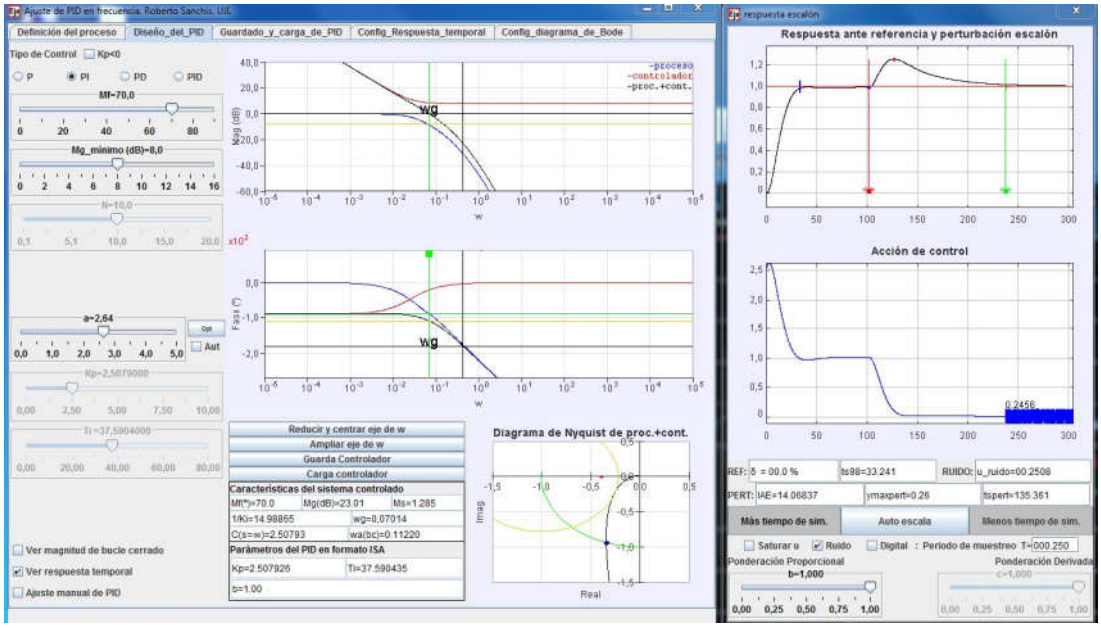



*Figura 37. Ajuste PID B24V + CAU + SENSOR*

Los resultados para el PI resultante son: Kp = 2.5079, Ti = 37.5904, b = 1, a = 0.05, N = 10, Mf = 70 y Mg = 8.

Al obtener el PID, se introducen los datos del mismo en la aplicación **PID_Arduino** para poder comprobar el correcto funcionamiento del mismo. Los resultados obtenidos son los siguientes:

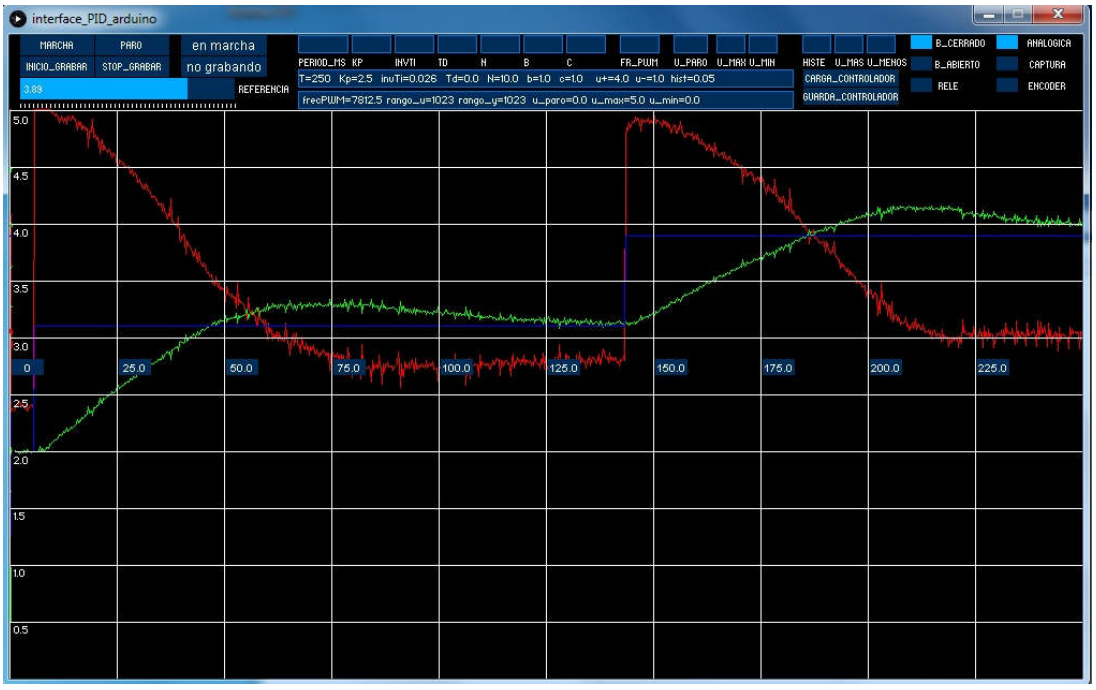



*Figura 38. DATOS Diseño PID aumento de referencia*

Como se puede observar en las gráficas, una vez se realizan los ensayos en bucle cerrado, es que, al aumentar la referencia, le cuesta más ajustarse debido a que la salida de agua durante las pruebas, va aumentado a medida que se aumenta el nivel en el depósito, ya que la presión aumenta.

Al realizar las pruebas disminuyendo la referencia, se ajusta bastante mejor sin llegar a sobrepasar tanto la referencia.

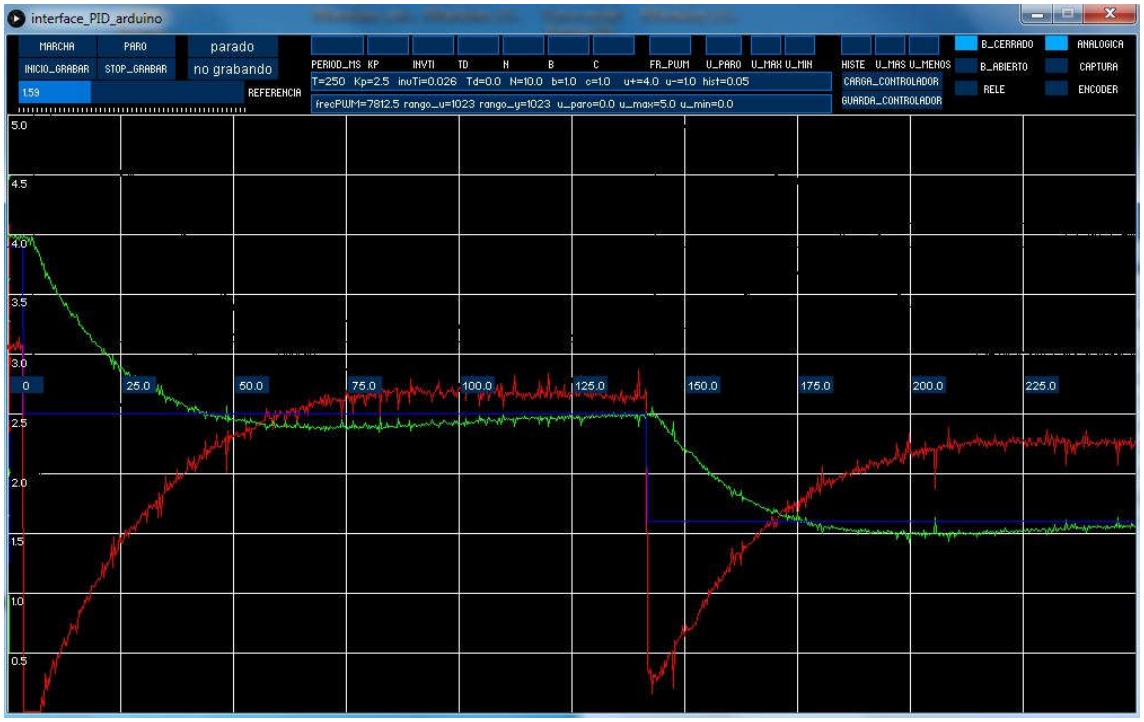



*Figura 39. DATOS Diseño PID aumento de referencia*

Con todas las comprobaciones realizadas y los bucles de control ajustados, se pasará a realizar todas las conexiones eléctricas con el PLC y la utilización de la aplicación CodeSys 2.3 de la compañía WAGO para la creación de un programa que pueda a llegar a controlar los distintos sensores y actuadores de manera automática a partir de la configuración de los distintos POU's.

## 1.7. Configuración y programación del PLC

La realización de las siguientes pruebas, es opcional, ya que, con el Arduino y todos los datos obtenidos, se puede realizar el estudio que se pretende obtener. Al usar el PID, o que se consigue es realizar todas las tareas de forma automática para no tener que realizar los ensayos de manera manual.

Para empezar, se deberá obtener todos los módulos necesarios para poder realizar todas las conexiones correctamente. Una vez obtenidos, el siguiente paso consiste en realizar todas las conexiones para que el PLC funcione correctamente. Una vea realizados, con la ayuda de un cable USB con el que el PLC se conecta al ordenador mediante la aplicación CodeSys v2.3, para así poder cargarle el programa, deberemos configurar la aplicación para que se pueda comunicar correctamente. Para ello se deben de realizar unos pasos para poder realizar esa configuración.

El primer paso consiste en crear un nuevo POU y seleccionar el modelo de PLC que se va a utilizar.

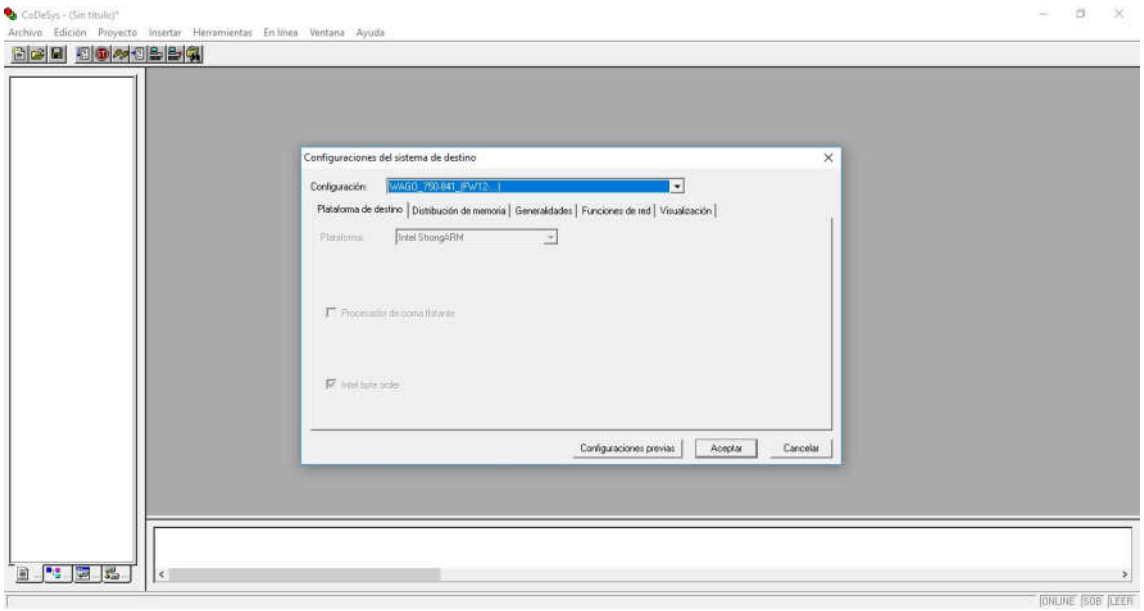



*Figura 40. Elección PLC*

En nuestro caso se elegirá la CPU 750-841 con la versión 12 de firmware.

A continuación, se deberán introducir los módulos que componen el PLC. Para ello el programa trae consigo una librería con todos los módulos, por lo que se deberán buscar los que disponemos y agregarlos al POU. En nuestro caso, hacen falta los módulos 750-468, 750-501, 750-550 y como final de línea el módulo 750-600.

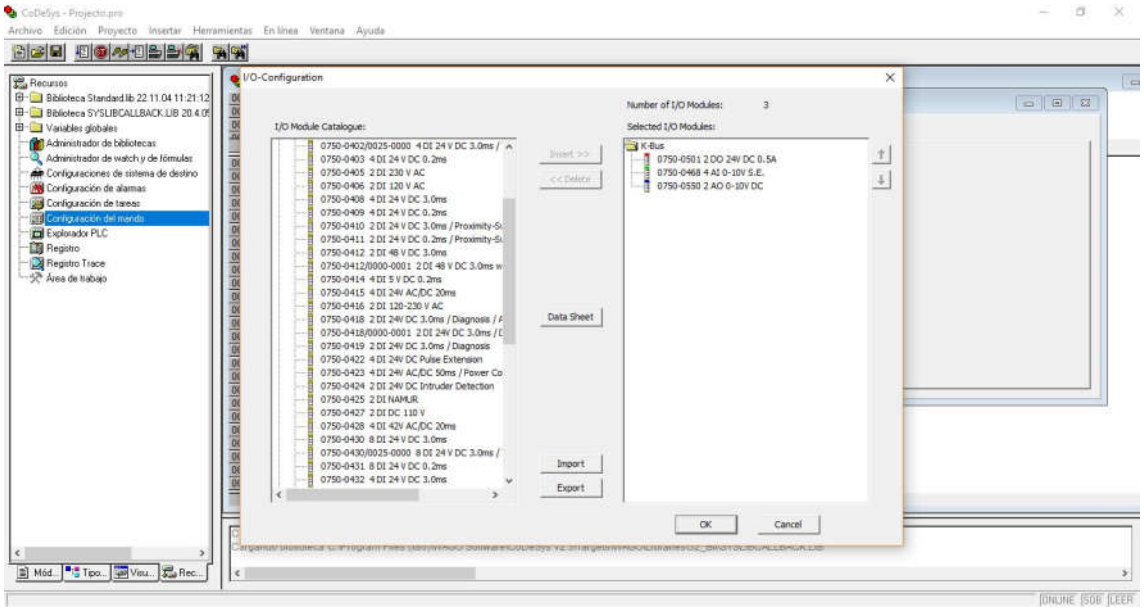



*Figura 41. Selección de módulos*

Al escoger los módulos, el programa directamente crea todas las variables necesarias.

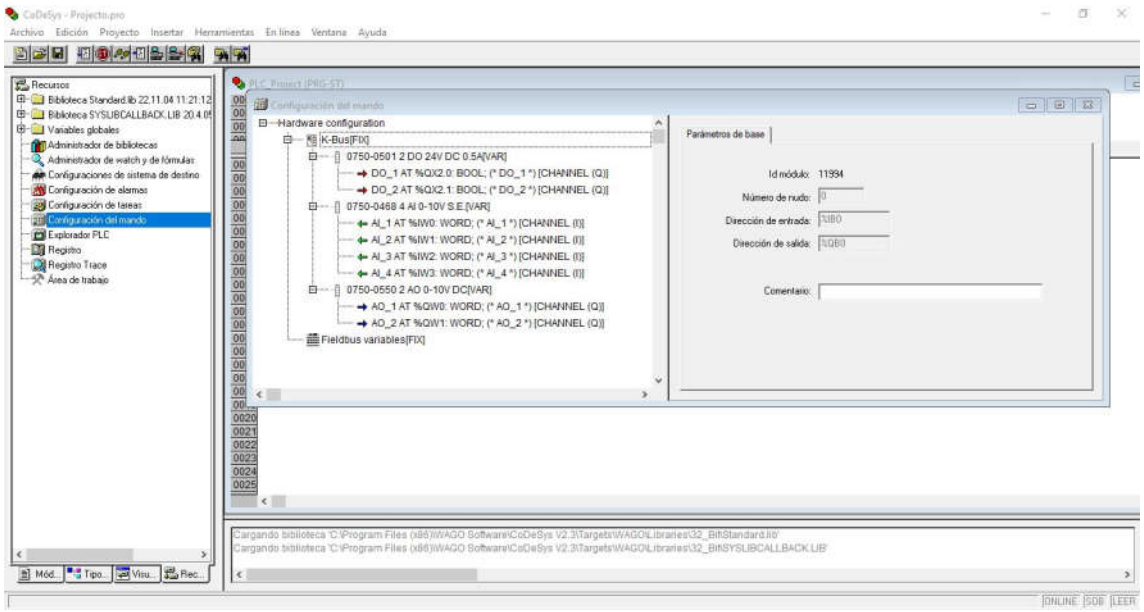



*Figura 42. Variables del sistema*

El último paso antes de diseñar el programa, es que el PLC se pueda comunicar con el PC. Por lo que con la ayuda del manual de WAGO, obtenemos la información necesaria para poder realizar correctamente esa conexión. Para ello, dentro del apartado comunicación, se deberá crear una nueva entrada para poder comunicarse por vía USB y seleccionar previamente, cual es el puerto USB que está utilizando el ordenador como puerto serie del PLC.

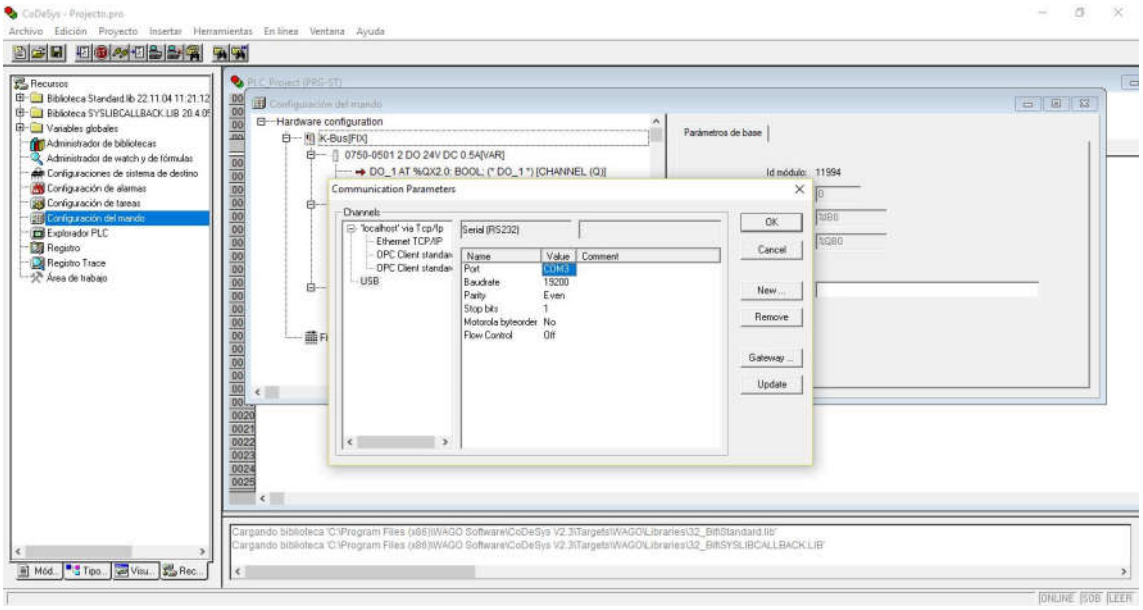



*Figura 43. Parámetros de conexión*

Una vez realizados y comprobados todos los pasos anteriores, se dispone a diseñar los distintos POU que compondrán al programa para un funcionamiento en concreto mediante el PLC.

Al tratarse de un sistema multivariable, se le pueden asignar distintas tareas dependiendo del uso que se quiera hacer con la instalación. En este caso, se realizará un estudio de llenado de los depósitos principales para el suministro de agua dependiendo de la demanda necesaria para el consumo con el fin de realizar el llenado de los depósitos cuando el precio hora de kW es más económico.

Para ello nos basaremos en una tarifa de electricidad para empresas 3.1 A, en la que el precio de la luz varía dependiendo la zona horaria en la que se encuentra. La zona de menor coste, se encuentra desde las 12 de la mañana hasta las 8 de la mañana. Las zonas intermedias de 8 de la mañana a 10 de la mañana y de 4 de la tarde a 12 de la noche. Y la zona más cara seria de 10 de la mañana a 4 de la tarde. Por lo que el llenado principal se realizará cuando se encuentre en la zona valle o zona más barata, en el que la demanda de agua es menor ya que el consumidor a estas horas se encuentra descansado. A continuación, cuando la demanda de agua aumenta, a la hora en la que el consumidor empieza su día laboral y se prepara para ir a trabajar, mantendremos en el nivel de agua del depósito a pesar de estar en una zona intermedia de consumo eléctrico regulando el llenado del depósito, aunque el caudal de salida aumente. Una vez llegamos al momento en el que la electricidad es más cara, lo que se hará es intentar no bombear agua al depósito principal e intentar que el depósito de agua llegue casi a su nivel mínimo para así no bombear en las horas punta de consumo eléctrico y producir un ahorro energético. Una vez

terminado el horario de consumo elevado, vuelve a un precio intermedio por lo que, en este horario, un horario en el que la demanda de agua vuelve a aumentar sin llegar a ser tan elevado como en la zona cara. En ese momento lo que se intentará es que el nivel del agua no llegue al mínimo, por lo que se llenará el depósito un poco por encima del nivel mínimo para que siempre exista una reserva de agua para no estropear las bombas y que se pueda suplir la demanda de agua en ese horario, y llegando a finalizar esta última franja horaria con el nivel del depósito casi al mínimo para así una vez volver a estar en horas valle, aprovechar para volver a llenar los depósitos de agua.

Para todo ello, como el sistema es multivariable, en este caso los materiales a utilizar, serán los siguientes:

- 2 bombas de agua
- 2 caudalímetros
- 1 sensor de nivel

La manera con la que se realizarán las conexiones es la siguiente. Para poder suministrar el agua al depósito y a su vez saber en todo momento la cantidad de agua que está entrando en él, se utilizará una bomba en serie con un caudalímetro. Para el control de nivel, se utilizará el sensor de nivel capacitivo obtenido para la realización del proyecto, que se comunicará con el caudalímetro y la bomba en cascada. Y, para la salida de agua, se volverá a utilizar otra bomba en serie con otro caudalímetro, que ira otra vez directa a los depósitos de acumulación para que siempre haya agua en los depósitos y no llegar a estropear las bombas.

Por lo que una vez configurado el programa de WAGO y establecido el proceso que va a seguir, se dispone a crear distintos POU's en el programa con el que este pueda realizar las funciones de una manera eficiente y correcta.

El primer paso consiste en crear la tarea principal, en la que como se podrá observa, es bastante sencilla. En este caso, el POU lo escribiremos en bloques (grafcet). Lo que hará es llenar el depósito al máximo en el caso de que no se encuentre lleno y dentro del *step2,* introduciremos los bloques de PID en cascada para que en todo momento realice los cálculos para el llenado de los depósitos.

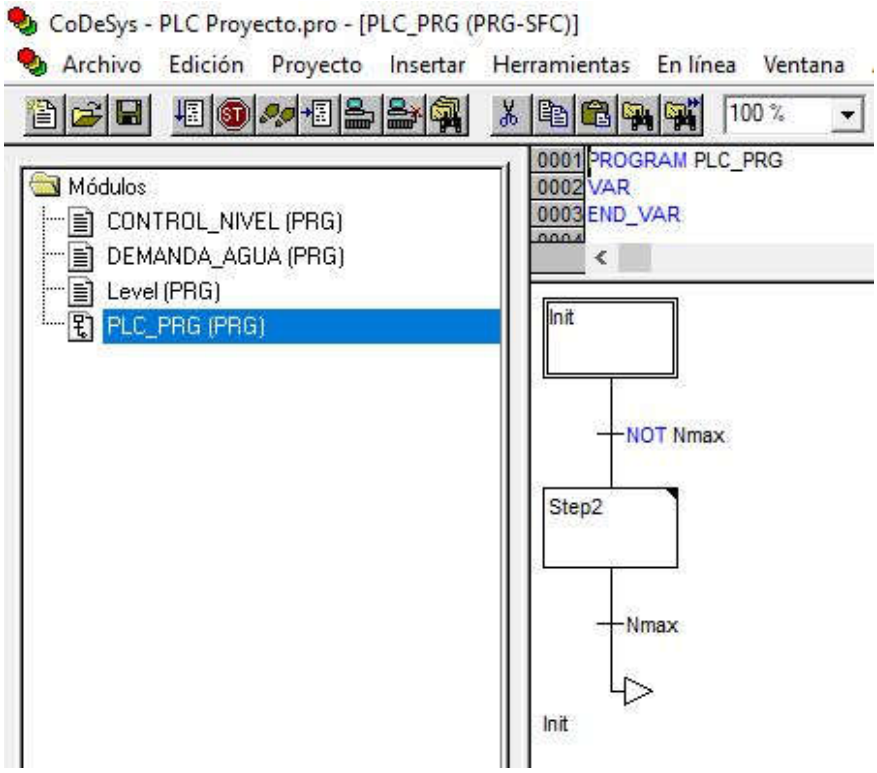



*Figura 44. Diseño PLC 1*

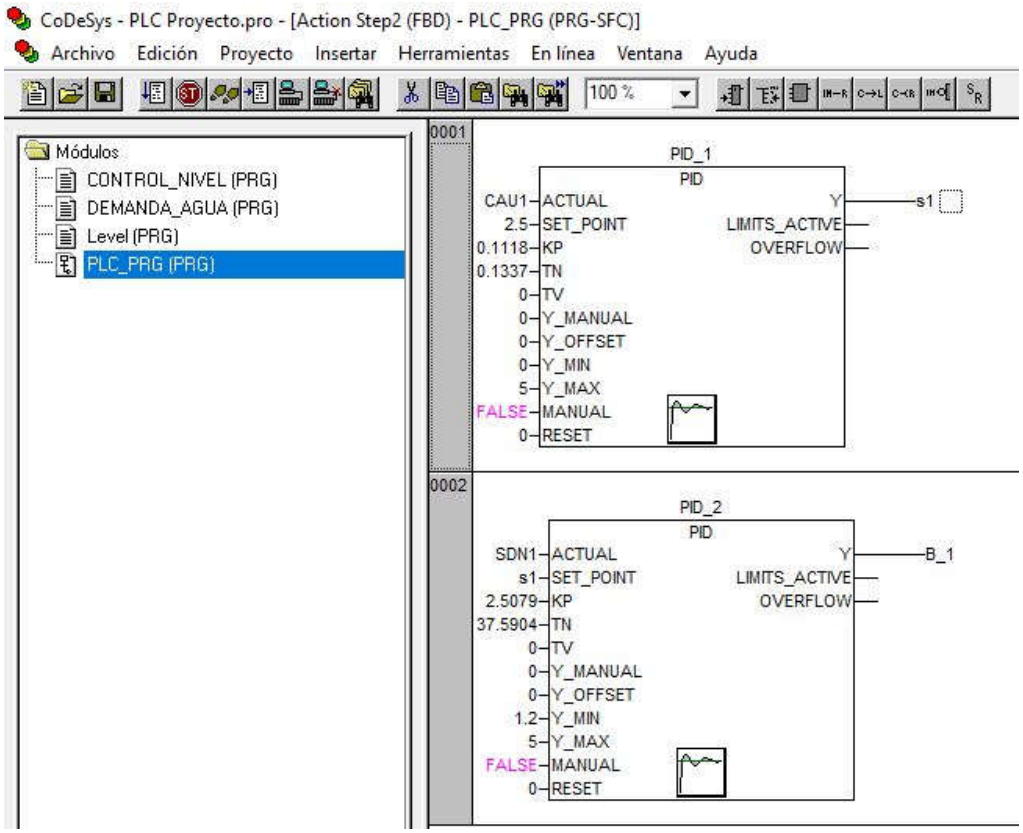



*Figura 45. Diseño PLC 2*

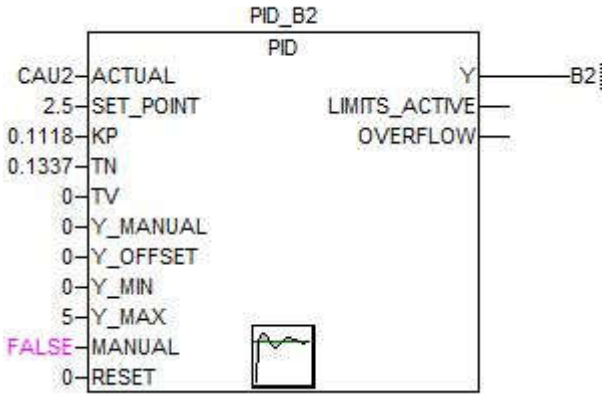



*Figura 46. Diseño PID_B2*

A continuación, para poder realizar el control respecto al tiempo, se crea una rutina en la que, con el valor del tiempo transcurrido, este calcula por una parte el nivel de llenado del depósito y con otro POU, la salida del agua dependiendo de la demanda exigida.

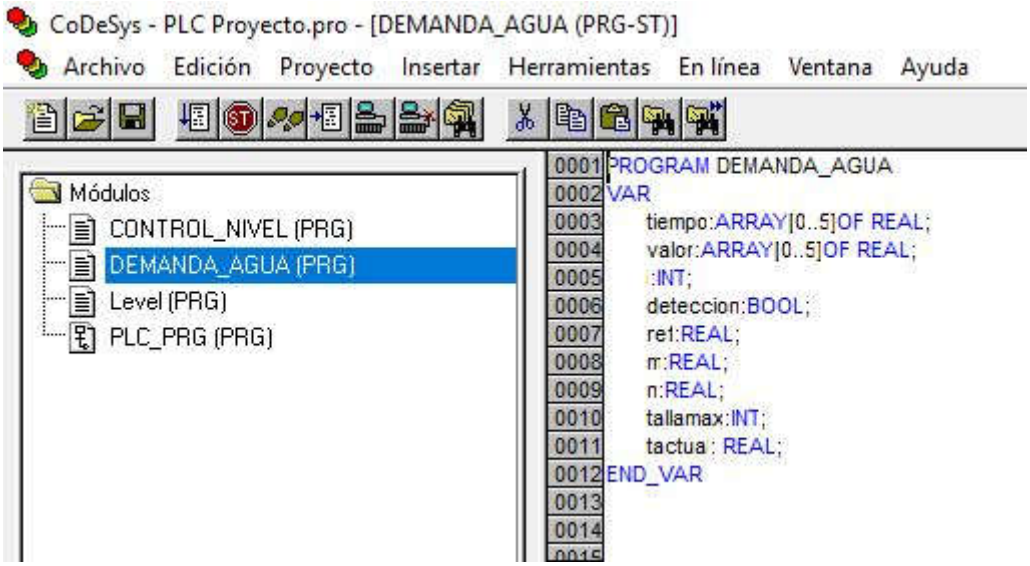



*Figura 47. Diseño PLC 3*

En la primera imagen, se muestran las variables creadas para este POU y a continuación el código para que realice su función. Lo que se ha introducido es una ecuación de una recta en la que, entre los valores preestablecidos, calcula el valor del nivel de referencia del depósito, interpolando cada cierto tiempo entre los distintos valores de array. Aquí será donde se establecerán las zonas horarias para cada tipo de precio eléctrico y donde se estimará la demanda de consumo de agua.

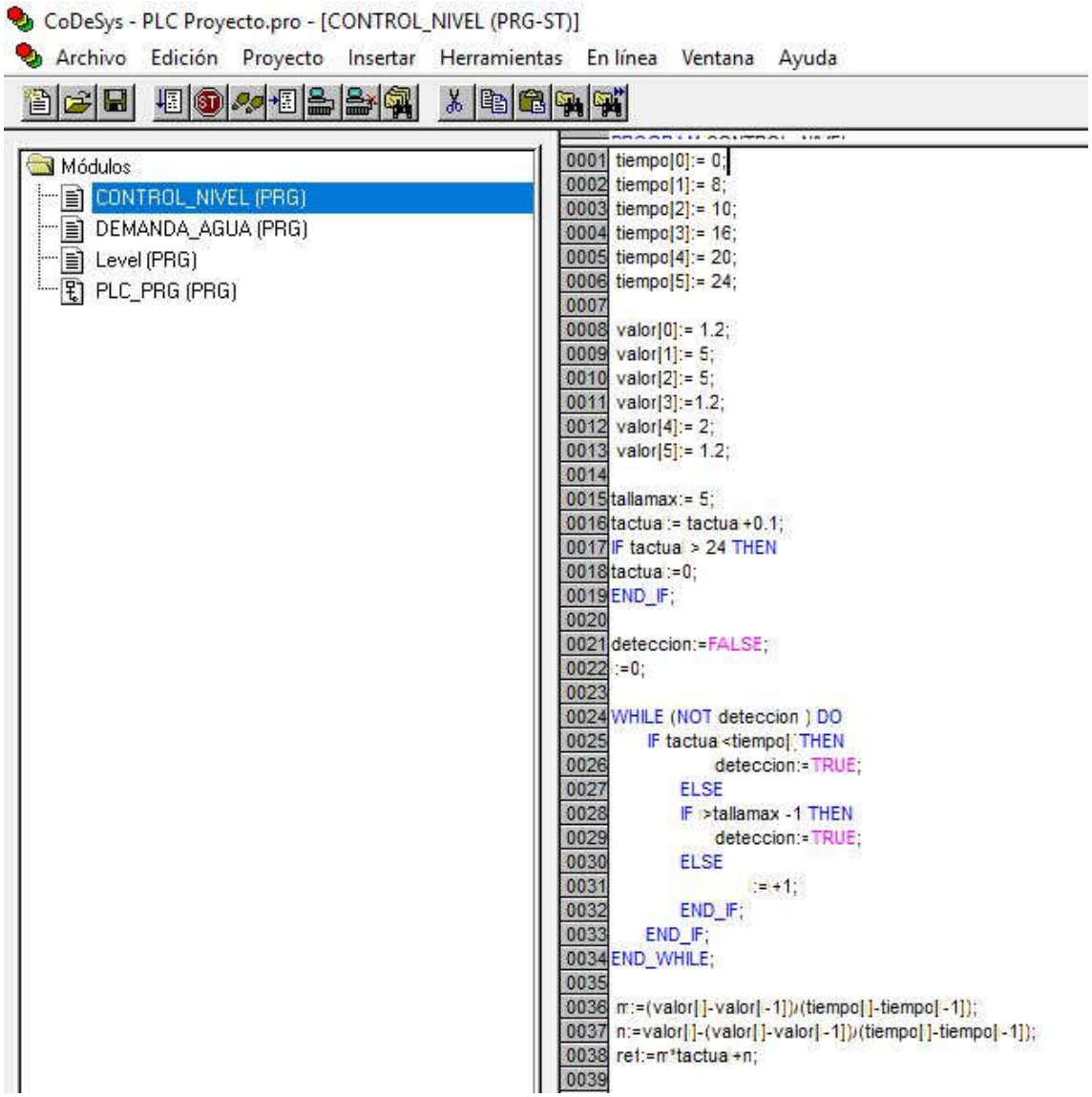



*Figura 48. Diseño PLC 4*

Igual que para controlar el nivel del depósito, se realiza otra igual, pero con los valores para la demanda de salida de agua.

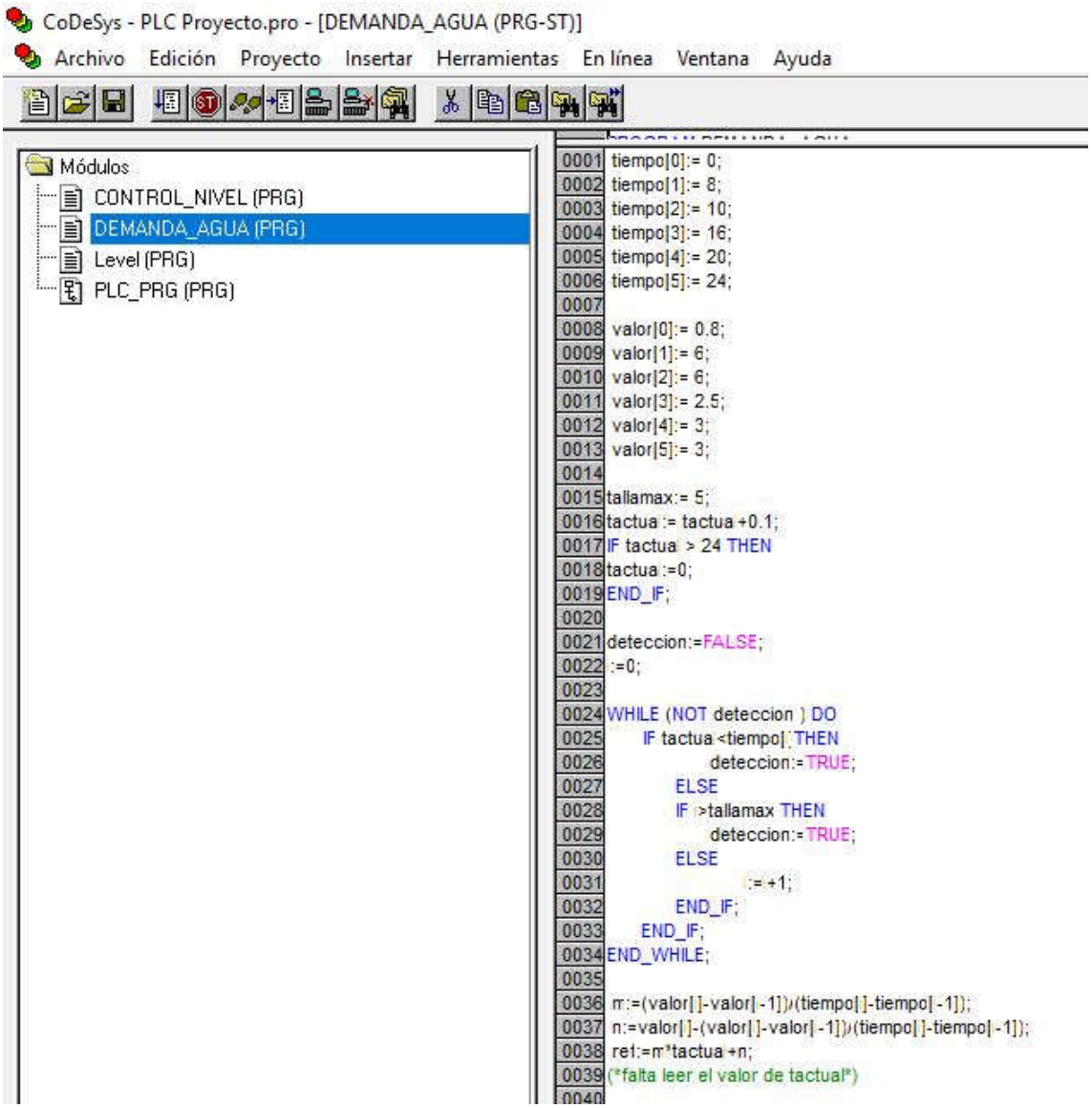



*Figura 49. Diseño PID 4*

Y así, como las distintas variables utilizadas durante el proyecto, ya que numerosas de ellas se han establecido como variables globales del sistema.

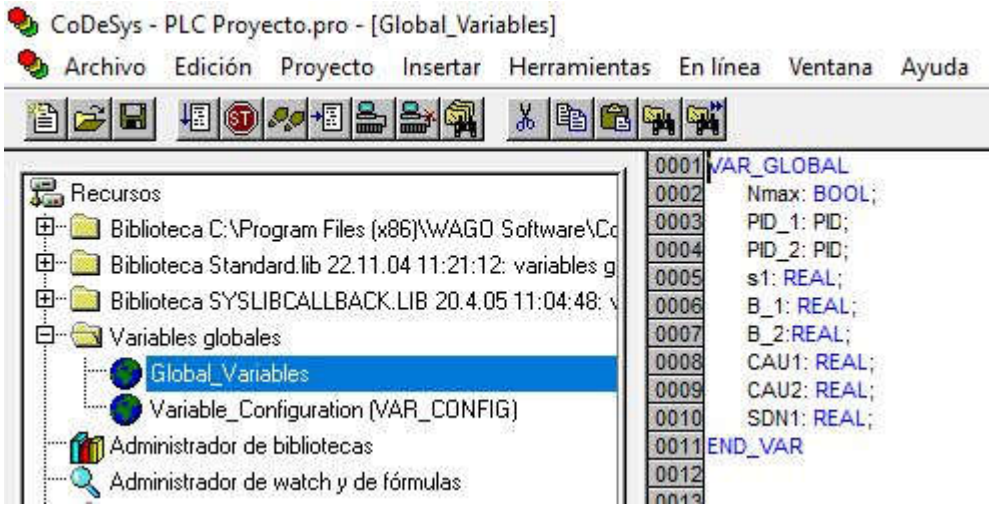



*Figura 50. Diseño PLC 5*

Para finalizar, los distintos cambios realizados a las variables de los módulos, ya que estas ya venían predefinidas y se debían cambiar.

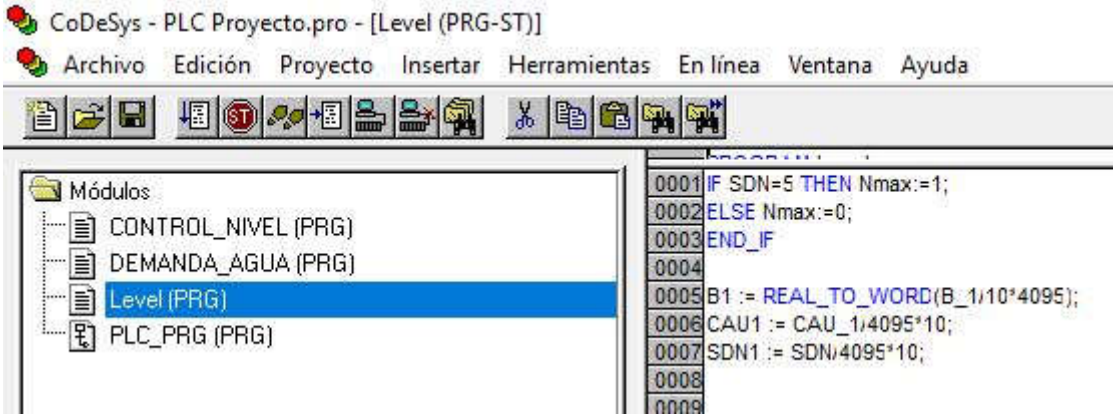



*Figura 51. Diseño PLC 6*

A partir del programa realizado, se llega a la descripción descrita anteriormente obteniendo los datos de manera correcta y verificando que realiza la función que se ha descrito

## 1.8. Viabilidad

### 1.8.1. Viabilidad técnica

Todos los dispositivos y componentes utilizados durante la realización del proyecto, están disponibles en el mercado actual, por lo que el proyecto se considera viable. Además, el estudio que se ha desarrollado con los distintos sensores y curvas de respuesta ha demostrado la viabilidad del equipo como simulador a escala de un sistema de bombeo real.

### 1.8.2. Viabilidad económica.

Comparado con el coste de un equipo comercial de laboratorio de este tipo (unos 30000 euros). El presupuesto de este proyecto resulta bastante más económico a relación con el coste de uno comercial. Por lo que en cuanto a viabilidad económica supera con creces los resultados esperados.

# 2. Pliego de condiciones

## 2.1. Ordenador

Requisitos mínimos que tiene que cumplir el ordenador para la utilización de los distintos programas usados durante el proyecto:

- Procesador (CPU): 2 o más núcleos con una frecuencia superior a 2 GHz
- Memoria de acceso aleatorio (RAM): 2GB o más de memoria
- Tarjeta gráfica (GPU): 512MB o superior
- Disco duro (HDD): 30GB de espacio disponible
- Sistema operativo (OS): Windows 7 64bits o versión superior
- Última versión Java disponible

## 2.2. Arduino

Arduino UNO R3 es una placa microcontroladora basada en ATmega328P. Tiene 14 pines digitales de entrada / salida (de los cuales 6 se pueden usar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reinicio. Contiene todo lo necesario para soportar el microcontrolador.

- Microcontrolador: ATmega328P
- Operating voltage: 5V
- Input voltaje (recomendado): 7-12V
- Input voltaje (máximo): 6-20V
- Digital I/O Pins: 14 (de los cuales, 6 salidas PWM)
- Analog Input Pins: 6
- DC current per I/O Pin: 20 mA
- DC current for 3.3V Pin: 50 mA
- Flash Memory: 32KB de los cuales 0.5KB usados para el bootloader
- SRAM: 2KB
- EEPROM: 1KB
- Clock Speed: 16MHz
- LED_BUILTIN: 13
- Medidas: (68.6x53.4) mm y 25gramos

## 2.3. CodeSys V2.3

La versión de CodeSys utilizada es muy similar a la versión V3.5. En concreto, esta versión es la utilizada por la compañía WAGO, ya que contiene todas las librerías necesarias para realizar cualquier operación con todos sus módulos y posteriormente crear une pequeña simulación del mismo.

## 2.4. PLC

El PLC a utilizar durante la realización del proyecto ha de ser de la compañía WAGO, ya que sino los programas utilizados no funcionarían y no se comunicarán correctamente con el ordenador.

## 2.5. Amplificador

El amplificador debe soportar al menos una tensión de 24V y una corriente de 2A. En este caso, el utilizado supera con creces estos valores. Además de llevar consigo una entrada analógica para el control de los motores de continua.

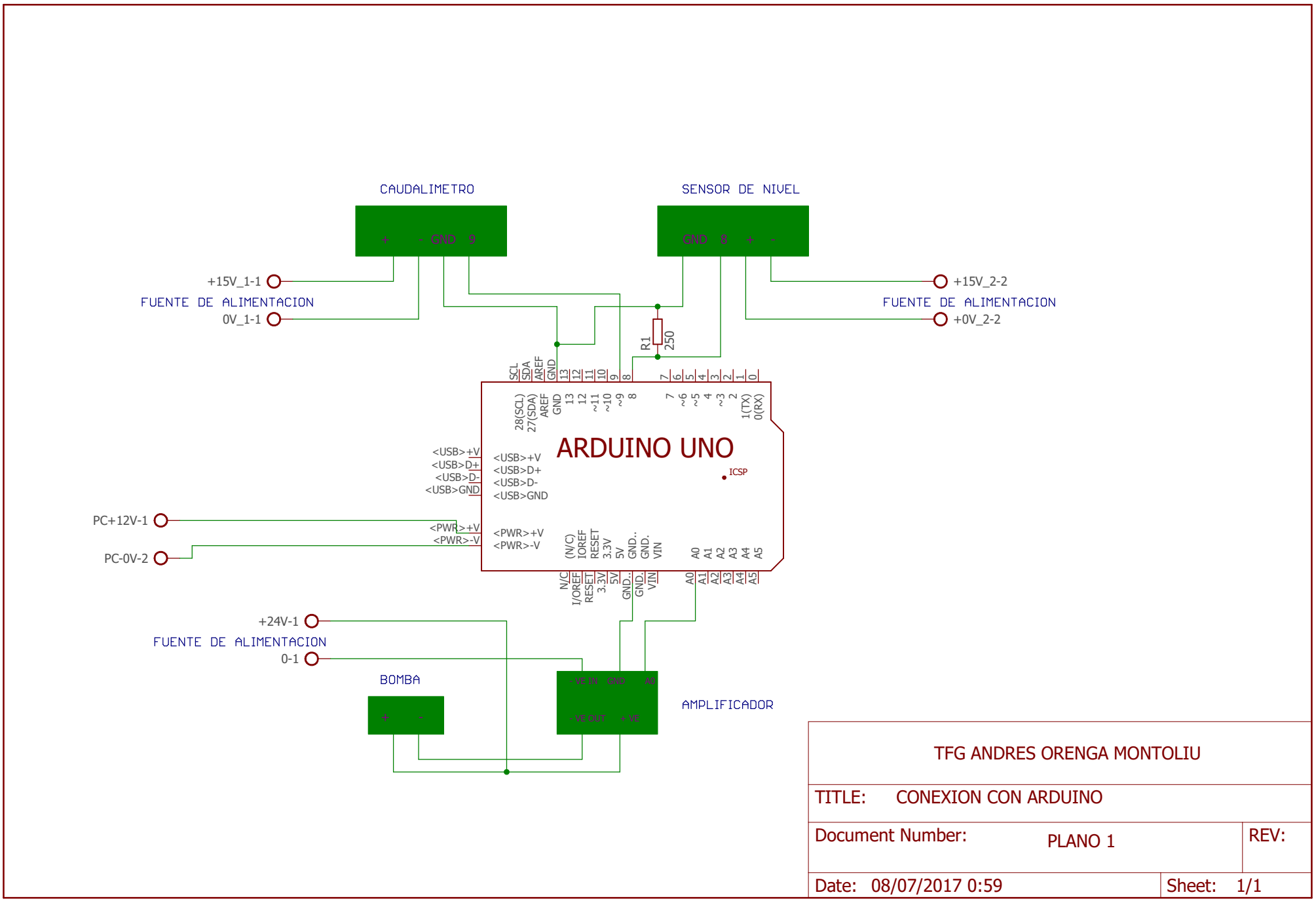
## 2.6. Caudalímetro

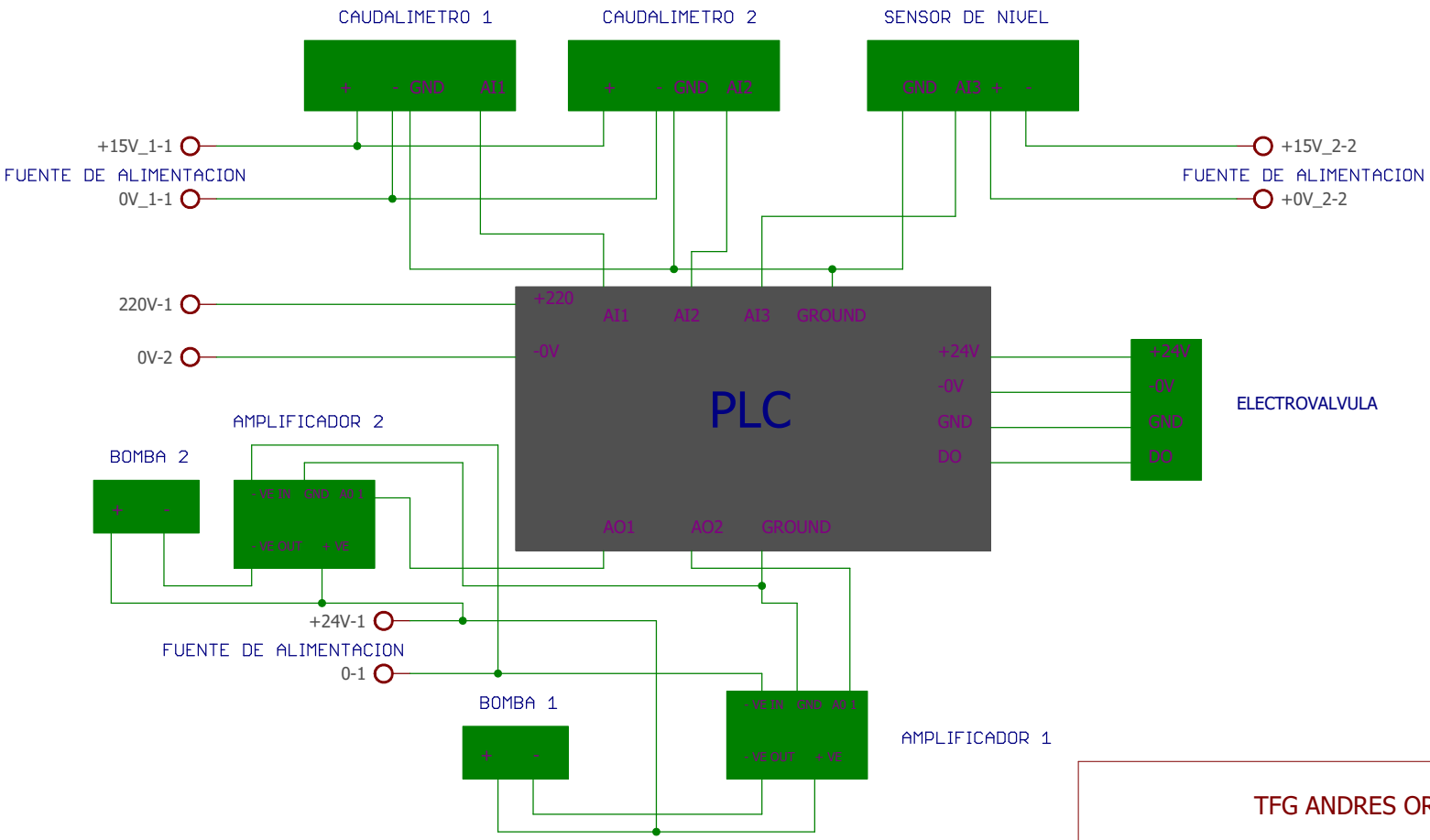
Debe contar con una baja caída de presión, tener un rango de medidas entre 0-10 l/min y además contar con una salida analógica con la que obtener los datos de medida independientemente del tipo de sensor a elección.

## 2.7. Sensor de nivel

El rango de medida del sensor debe de ser al menos de 50cm para poder tomar las medidas de los depósitos utilizados en el proyecto y además como en el resto de los componentes una salida analógica con la que obtener los datos de nivel.

# 3. Planos


CAUDALIMETRO
+ - GND 9
SENSOR DE NIVEL
GND 8 + -
+15V_1-1
FUENTE DE ALIMENTACION
0V_1-1
+15V_2-2
FUENTE DE ALIMENTACION
+0V_2-2
R1
250
ARDUINO UNO
ICSP
<USB>+V
<USB>D+
<USB>D-
<USB>GND
<PWR>+V
<PWR>-V
PC+12V-1
PC-0V-2
+24V-1
FUENTE DE ALIMENTACION
0-1
BOMBA
+ -
- VE IN GND A0
- VE OUT + VE
AMPLIFICADOR
TFG ANDRES ORENGA MONTOLIU
TITLE: CONEXION CON ARDUINO
Document Number: PLANO 1
REV:
Date: 08/07/2017 0:59
Sheet: 1/1




| | | |
|---|---|---|
| TFG ANDRES ORENGA MONTOLIU | | |
| TITLE: CONEXION CON PLC | | |
| Document Number: PLANO 2 | | REV: |
| Date: 08/07/2017 0:55 | Sheet: 1/1 | |

# 4. Presupuesto

## COMPONENTES ADQUIRIDOS PARA EL PROYECTO

| Componentes | Unidades | Precio unitario (€) | Precio total (€) |
|---|---|---|---|
| Caudalímetro | 4 | 210 | 840 |
| Sensor de nivel | 2 | 370 | 740 |
| Bomba 24V | 4 | 19.9 | 79.6 |
| Electroválvula | 2 | 19.9 | 38.8 |
| Amplificador | 4 | 57.18 | 228.72 |
| Componentes varios | | | 10 |

Tabla 3. Componentes proyecto

## SISTEMA DE DEPÓSITOS

Los depósitos ya se encuentran en el laboratorio, pero el diseño de la instalación tuvo un coste de 1500€ para los dos depósitos.

## COMPONENTES ADQUISICIÓN DE DATOS

| Componentes | Unidades | Precio unitario (€) | Precio total (€) |
|---|---|---|---|
| PCL WAGO 750-841 | 1 | 340.82 | 340.82 |
| Módulo 750-501 | 1 | 42.82 | 42.82 |
| Módulo 750-468 | 1 | 37.53 | 37.53 |
| Módulo 750-550 | 1 | 250.36 | 250.36 |
| Módulo 750-600 | 1 | 23.68 | 23.68 |
| Arduino UNO R3 | 1 | 25.41 | 25.41 |

Tabla 4. Componentes adquisición de datos

## HORAS PERSONAL

| Profesional | Horas | Precio unitario (€) | Precio total (€) |
|---|---|---|---|
| Operario | 10 | 25 | 250 |
| Ingeniero industrial | 150 | 40 | 6000 |

Tabla 5. Horas personal

## PRESUPUESTO FINAL

El presupuesto final para la realización de todo el proyecto, es de 10407,74 euros

# 5. Anexos

## 5.1. Programación Arduino bomba y caudalímetro

```
/*
  Analog input, analog output, serial output

 Reads an analog input pin, maps the result to a range from 0 to 255
 and uses the result to set the pulsewidth modulation (PWM) of an output pin.
 Also prints the results to the serial monitor.

 CODIGO BOMBA CON CAUDALIMETRO

 created 29 Dec. 2008
 modified 12 Apr 2017 by Andrés Orenga
 by Tom Igoe

 This example code is in the public domain.

 */

#include <FlexiTimer2.h>
#define encoderPinA 2
#define encoderPinB 3
#define encoderMax 3500
#define encoderMin -1000

const byte counterPin = 3, counterInterrupt = 1;
const int analogInPin = A0, analogOutPin = 9;


int sensorValue = 0, outputValue = 0, interval=10, uparo=511, umas=818, umenos=205, contador=0;
int u_max=1023, umax=1023, u_min=0, ymax=2400, refint;
long previousMillis, cuentas=0, cuentasant=0;
float I, D, ref=512.0;
char preescalado=1, mp, * numero;
byte refb1, refb2;
float e, refant=500, yant=0, y, u=0, T=0.01, Tim1=0, Kp=0, Td=0, N=20.0, b=1, c=1, hist=10, refe, frec=0;
int marchaparo=0, periodocaptura=0;
boolean actualfrec=false, bucleabierto=true, enviadatos=false, rele=false, A_set, B_set;
int entrada=1, encoderPos=0; // 1=analogica, 2=captura, 3=encoder

void flash(){
    if ( marchaparo==0){
        outputValue=(int) (uparo*1.0*umax/1023.0);
          if (outputValue>umax) outputValue=umax;
          if (outputValue<0) outputValue=0;
              analogWrite(analogOutPin, outputValue);
                      }
          else{
                enviadatos=true;
              }
           }
void setup() {
  TCCR1A = 0x00; // sets timer control bits to PWM Phase and Frequency Correct mode
  TCCR1B = 0x11; // sets timer control bits to Prescaler N = 1
  ICR1 = 0x0400; // 8Khz, 10 bits
```

```
// initialize serial communications at 9600 bps:
Serial.begin(115200);
//attachInterrupt(counterInterrupt, counterISR, CHANGE);
    pinMode(encoderPinA, INPUT);
    pinMode(encoderPinB, INPUT);

   FlexiTimer2::set(interval, flash);
   FlexiTimer2::start();
          }

void loop() {
    if (entrada==2) {
    if (actualfrec==true){
        frec=(frec*(8192.0-periodocaptura)+62830.0)/8192.0;
        sensorValue=int(frec);
        actualfrec=false;
                        }
                    }
    if (enviadatos==true){
    if (entrada==2)   sensorValue=int(frec);
    if (entrada==1) sensorValue = analogRead(analogInPin);
    if (entrada==3) sensorValue=encoderPos;
    if (bucleabierto==true){
    if (outputValue>umax) outputValue=umax;
    if (outputValue<0) outputValue=0;


 analogWrite(analogOutPin, outputValue);
 Serial.write(highByte(outputValue));
 Serial.write(lowByte(outputValue));
 Serial.write(highByte(sensorValue));
 Serial.write(lowByte(sensorValue));
 Serial.write(highByte(int(ref)));
 Serial.write(lowByte(int(ref)));
                    }
else {
    if (entrada==3) {
        y=float(sensorValue*1023.0/((float) ymax));
        refe=ref*1023.0/((float) ymax);
                }
        else {
             y = float(sensorValue);
             refe=ref;
            }
             if (rele==false){
             if (Kp>0) {
             if (((u>u_min)||(refe>y))&&((u<u_max)||(refe<y))) I=I+Kp*T*Timl*(refe-y);
                    }
                    else {
                         if (((u>u_min)||(refe<y))&&((u<u_max)||(refe>y))) I=I+Kp*T*Timl*(refe-y);
                        }
```

```
                        if (Timl==0) I=0;
                        if (Td==0) D=0;
                        else  D=(Td*D+Kp*Td*N*(c*refe-y-c*refant+yant))/(N*T+Td);
                              u=Kp*(b*refe-y)+I+D+(uparo*1023.0)/umax;
                        if (u>u_max) u=u_max;
                        if (u<u_min) u=u_min;
                            yant=y;
                            refant=refe;
                        }
                        else {
                              if ((u!=umenos)&&(y>(refe+hist)))  u=umenos;
                              if ((u!=umas)&&(y<(refe-hist)))  u=umas;
                             }
                              outputValue=int((u*umax)/1023);
                              analogWrite(analogOutPin, outputValue);
                              Serial.write(highByte(outputValue));
                              Serial.write(lowByte(outputValue));
                              Serial.write(highByte(sensorValue));
                              Serial.write(lowByte(sensorValue));
                              Serial.write(highByte(int(ref)));
                              Serial.write(lowByte(int(ref)));
}
 enviadatos=false;
                    }
```

```
mp=Serial.read();
if(mp=='p'){
   marchaparo=0;
            }
if (mp=='m'){
    marchaparo=1;
             }
if(mp=='s'){
   while (Serial.available()<2) ;
          if (bucleabierto==false) {
              ref=(float) ((Serial.read()<<8) | (Serial.read()));
                                  }
              else {
                    outputValue=(int) ((Serial.read()<<8) | (Serial.read()));
                   }
            }
if(mp=='t'){
   while (Serial.available()<2) ;
          interval=(int) ((Serial.read()<<8) | (Serial.read()));
          T=float(interval/1000.0);
          FlexiTimer2::stop();
          FlexiTimer2::set(interval, flash);
          FlexiTimer2::start();
           }
```

```
if(mp=='y'){
            while (Serial.available()<2) ;
                  ymax=(int) ((Serial.read()<<8) | (Serial.read()));
            }
if(mp=='i'){
   while (Serial.available()<4) ;
          int Timli=((Serial.read()<<8)|(Serial.read()));
          int Tilmd=((Serial.read()<<8)|(Serial.read()));
          Timl=(float) (Timli+Tilmd/10000.0);
           }
if(mp=='k'){
   while (Serial.available()<4) ;
          int Kpi=((Serial.read()<<8)|(Serial.read()));
          int Kpd=((Serial.read()<<8)|(Serial.read()));
          Kp=(float) ((float) Kpi+((float) Kpd)/10000.0);
           }
if(mp=='d'){
   while (Serial.available()<4) ;
          int Tdi=((Serial.read()<<8)|(Serial.read()));
          int Tdd=((Serial.read()<<8)|(Serial.read()));
          Td=(float) (Tdi+Tdd/10000.0);
           }
if(mp=='n'){
   while (Serial.available()<4) ;
          int Ni=((Serial.read()<<8)|(Serial.read()));
          int Nd=((Serial.read()<<8)|(Serial.read()));
          N=(float) (Ni+Nd/10000.0);
```

```
                }
        if(mp=='b'){
           while (Serial.available()<4) ;
                  int bi=((Serial.read()<<8)|(Serial.read()));
                  int bd=((Serial.read()<<8)|(Serial.read()));
                  b=(float) (bi+bd/10000.0);
                  }
        if(mp=='c'){
           while (Serial.available()<4) ;
                  int ci=((Serial.read()<<8)|(Serial.read()));
                  int cd=((Serial.read()<<8)|(Serial.read()));
                  c=(float) (ci+cd/10000.0);
                  }
        if(mp=='u'){
           while (Serial.available()<2) ;
                  uparo=(int) ((Serial.read()<<8) | (Serial.read()));
                  }
        if(mp=='h'){
           while (Serial.available()<4) ;
                  u_max=(int) ((Serial.read()<<8) | (Serial.read()));
                  u_min=(int) ((Serial.read()<<8) | (Serial.read()));
                  }
        if(mp=='j'){
           while (Serial.available()<2) ;
                  umas=(int) ((Serial.read()<<8) | (Serial.read()));
                  }
```

```
if(mp=='f'){
   while (Serial.available()<3) ;
          preescalado=Serial.read();
          umax=(int) ((Serial.read()<<8) | (Serial.read()));
          TCCR1B = 0x10+preescalado; // sets timer control bits to Prescaler N = 1
          ICR1 = umax+1;//0x0400; // 8Khz, 10 bits
           }
if (mp=='a'){
    bucleabierto=true;
    rele=false;
           }
if (mp=='l'){
    bucleabierto=false;
    rele=false;
           }
if (mp=='z'){
    bucleabierto=false;
    rele=true;
    u=umenos;
           }
if (mp=='g'){
    entrada=1;
    detachInterrupt(0);
    detachInterrupt(1);
           }
```

```
                }
        if (mp=='o'){
            entrada=2;
            detachInterrupt(1);
                }
        if (mp=='e'){
            entrada=3;
                }
        if(mp=='w'){
           while (Serial.available()<2) ;
                  hist=(float) ((Serial.read()<<8) | (Serial.read()));
                  }
                }
```

## 5.2. Programación Arduino sensor de nivel

```
/*
  Analog input, analog output, serial output

 Reads an analog input pin, maps the result to a range from 0 to 255
 and uses the result to set the pulsewidth modulation (PWM) of an output pin.
 Also prints the results to the serial monitor.

CODIGO SENSOR DE NIVEL

 created 29 Dec. 2008
 modified 12 Apr 2017 by Andrés Orenga
 by Tom Igoe

 This example code is in the public domain.

 */

#include <FlexiTimer2.h>
#define encoderPinA 2
#define encoderPinB 3
#define encoderMax 3500
#define encoderMin -1000

const byte counterPin = 3, counterInterrupt = 1;
//FreqPeriodCounter counter(counterPin, micros, 0);
// These constants won't change.  They're used to give names
// to the pins used:
```

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogInPincaudal = A1;
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0, outputValue = 0, interval=10, uparo=511, umas=818, umenos=205, contador=0;
int u_max=1023, umax=1023, u_min=0, ymax=2400, refint;
long previousMillis, cuentas=0, cuentasant=0;
float I, D, ref=512.0, yC, redC, uC, IC, KpC=0.1118, TimlC=7.479, bC=1;
char preescalado=1, mp, * numero;
byte refbl, refb2;
float e, refant=500, yant=0, y, u=0, T=0.01, Timl=0, Kp=0, Td=0, N=20.0, b=1, c=1, hist=10, refe, frec=0;
int marchaparo=0, periodocaptura=0;
boolean actualfrec=false, bucleabierto=true, enviadatos=false, rele=false, A_set, B_set;
int entrada=1, encoderPos=0; // 1=analogica, 2=captura, 3=encoder


void flash(){
    if ( marchaparo==0){
        outputValue=(int) (uparo*1.0*umax/1023.0);
          if (outputValue>umax) outputValue=umax;
          if (outputValue<0) outputValue=0;
              analogWrite(analogOutPin, outputValue);
                      }
          else{
                enviadatos=true;
              }
           }
```

```
void setup() {
  TCCR1A = 0x00; // sets timer control bits to PWM Phase and Frequency Correct mode
  TCCR1B = 0x11; // sets timer control bits to Prescaler N = 1
  ICR1 = 0x0400; // 8Khz, 10 bits
  // initialize serial communications at 9600 bps:
  Serial.begin(115200);
  //attachInterrupt(counterInterrupt, counterISR, CHANGE);
      pinMode(encoderPinA, INPUT);
      pinMode(encoderPinB, INPUT);

     FlexiTimer2::set(interval, flash);
     FlexiTimer2::start();
             }

void loop() {
    if (entrada==2) {
    if (actualfrec==true){
        frec=(frec*(8192.0-periodocaptura)+62830.0)/8192.0;
        sensorValue=int(frec);
        actualfrec=false;
                          }
                      }
  if (enviadatos==true){
    if (entrada==2)    sensorValue=int(frec);
    if (entrada==1)
    {
```

```
      sensorValue = analogRead(analogInPin);
      caudalValue = analogRead(analogInPincaudal);
    }
    if (entrada==3) sensorValue=encoderPos;
    if (bucleabierto==true){
        refC=float(refCaudalValue);
        yC = float(caudalValue);
        if (((uC>u_min)||(refC>yC))&&((uC<u_max)||(refC<yC))) IC=IC+KpC*T*TimlC*(refC-yC);
            uC=KpC*(bC*refC-yC)+IC;
            if (uC>u_max) uC=u_max;
            if (uC<u_min) uC=u_min;
                 outputValue=int((uC*umax)/1023);
                  if (outputValue>umax) outputValue=umax;
                  if (outputValue<0) outputValue=0;
                  analogWrite(analogOutPin, outputValue);
                  Serial.write(highByte(refCaudalValue));
                  Serial.write(lowByte(refCaudalValue));
                  Serial.write(highByte(sensorValue));
                  Serial.write(lowByte(sensorValue));
                  Serial.write(highByte(int(ref)));
                  Serial.write(lowByte(int(ref)));
                            }
else {
      if (entrada==3) {
          y=float(sensorValue*1023.0/((float) ymax));
          refe=ref*1023.0/((float) ymax);
                      }
```

```
        else {
              y = float(sensorValue);
              refe=ref;
             }
              if (rele==false){
              if (Kp>0) {
              if (((u>u_min)||(refe>y))&&((u<u_max)||(refe<y))) I=I+Kp*T*Timl*(refe-y);
                        }
                        else {
                              if (((u>u_min)||(refe<y))&&((u<u_max)||(refe>y))) I=I+Kp*T*Timl*(refe-y);
                             }
                              if (Timl==0) I=0;
                              if (Td==0) D=0;
                              else  D=(Td*D+Kp*Td*N*(c*refe-y-c*refant+yant))/(N*T+Td);
                                    u=Kp*(b*refe-y)+I+D+(uparo*1023.0)/umax;
                              if (u>u_max) u=u_max;
                              if (u<u_min) u=u_min;
                                  yant=y;
                                  refant=refe;
                              }
                               else {
                                    if ((u!=umenos)&&(y>(refe+hist)))  u=umenos;
                                    if ((u!=umas)&&(y<(refe-hist)))  u=umas;
                                    }
```

```
refCaudalValue=int((u*umax)/1023);
refC=float(refCaudalValue);
yC = float(caudalValue);
if (((uC>u_min)||(refC>yC))&&((uC<u_max)||(refC<yC))) IC=IC+KpC*T*TimlC*(refC-yC);
uC=KpC*(bC*refC-yC)+IC;
if (uC>u_max) uC=u_max;
if (uC<u_min) uC=u_min;
outputValue=int((uC*umax)/1023);
analogWrite(analogOutPin, outputValue);
Serial.write(highByte(outputValue));
Serial.write(lowByte(outputValue));
Serial.write(highByte(sensorValue));
Serial.write(lowByte(sensorValue));
Serial.write(highByte(int(ref)));
Serial.write(lowByte(int(ref)));
   }
enviadatos=false;
                        }
   mp=Serial.read();
   if(mp=='p'){
      marchaparo=0;
               }
   if (mp=='m'){
       marchaparo=1;
                }


   if(mp=='s'){
      while (Serial.available()<2) ;
             if (bucleabierto==false) {
                 ref=(float) ((Serial.read()<<8) | (Serial.read()));
                                      }
                 else {
                       outputValue=(int) ((Serial.read()<<8) | (Serial.read()));
                      }
              }
   if(mp=='t'){
      while (Serial.available()<2) ;
             interval=(int) ((Serial.read()<<8) | (Serial.read()));
             T=float(interval/1000.0);
             FlexiTimer2::stop();
             FlexiTimer2::set(interval, flash);
             FlexiTimer2::start();
              }
   if(mp=='y'){
               while (Serial.available()<2) ;
                     ymax=(int) ((Serial.read()<<8) | (Serial.read()));
              }
   if(mp=='i'){
      while (Serial.available()<4) ;
             int Timli=((Serial.read()<<8)|(Serial.read()));
             int Tilmd=((Serial.read()<<8)|(Serial.read()));
             Timl=(float) (Timli+Tilmd/10000.0);
              }
```

```
if(mp=='k'){
   while (Serial.available()<4) ;
          int Kpi=((Serial.read()<<8)|(Serial.read()));
          int Kpd=((Serial.read()<<8)|(Serial.read()));
          Kp=(float) ((float) Kpi+((float) Kpd)/10000.0);
           }
if(mp=='d'){
   while (Serial.available()<4) ;
          int Tdi=((Serial.read()<<8)|(Serial.read()));
          int Tdd=((Serial.read()<<8)|(Serial.read()));
          Td=(float) (Tdi+Tdd/10000.0);
           }
if(mp=='n'){
   while (Serial.available()<4) ;
          int Ni=((Serial.read()<<8)|(Serial.read()));
          int Nd=((Serial.read()<<8)|(Serial.read()));
          N=(float) (Ni+Nd/10000.0);
           }
if(mp=='b'){
   while (Serial.available()<4) ;
          int bi=((Serial.read()<<8)|(Serial.read()));
          int bd=((Serial.read()<<8)|(Serial.read()));
          b=(float) (bi+bd/10000.0);
           }
```

```
      if(mp=='c'){
         while (Serial.available()<4) ;
                int ci=((Serial.read()<<8)|(Serial.read()));
                int cd=((Serial.read()<<8)|(Serial.read()));
                c=(float) (ci+cd/10000.0);
                 }
      if(mp=='u'){
         while (Serial.available()<2) ;
                uparo=(int) ((Serial.read()<<8) | (Serial.read()));
                 }
      if(mp=='h'){
         while (Serial.available()<4) ;
                u_max=(int) ((Serial.read()<<8) | (Serial.read()));
                u_min=(int) ((Serial.read()<<8) | (Serial.read()));
                 }
      if(mp=='j'){
         while (Serial.available()<2) ;
                umas=(int) ((Serial.read()<<8) | (Serial.read()));
                 }
      if(mp=='q'){
         while (Serial.available()<2) ;
                umenos=(int) ((Serial.read()<<8) | (Serial.read()));
                 }
if(mp=='f'){
   while (Serial.available()<3) ;
          preescalado=Serial.read();
          umax=(int) ((Serial.read()<<8) | (Serial.read()));
          TCCR1B = 0x10+preescalado; // sets timer control bits to Prescaler N = 1
          ICR1 = umax+1;//0x0400; // 8Khz, 10 bits
           }
if (mp=='a'){
    bucleabierto=true;
    rele=false;
           }
if (mp=='l'){
    bucleabierto=false;
    rele=false;
           }
if (mp=='z'){
    bucleabierto=false;
    rele=true;
    u=umenos;
           }
if (mp=='g'){
    entrada=1;
    detachInterrupt(0);
    detachInterrupt(1);
           }
```

```
if (mp=='o'){
    entrada=2;
    detachInterrupt(1);
            }
if (mp=='e'){
    entrada=3;
            }
if(mp=='w'){
   while (Serial.available()<2) ;
          hist=(float) ((Serial.read()<<8) | (Serial.read()));
           }
         }
```

## 5.3. Caudalímetro

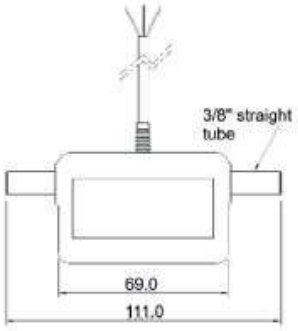
### *Ultrasonic Flow Meter UF08B*


This innovative design provides a high accuracy, non-invasive, flow measurement device at a fraction of the cost of other current non-invasive systems. The unique measurement technique automatically compensates for viscosity and temperature variations. The measurement of flow is by ultrasonic transit time in-line cell.

The flow path is designed to minimise pressure drop and, having no moving parts within, will not clog or jam. The sensor also allows contaminants to pass through without affecting its performance.

The sensor is supplied with pulse and analogue outputs, which are selectable by connection wires..

- Non-invasive sensor technology
- Fully electronic, no moving parts
- Automatic viscosity and temperature compensation
- Pulse or analogue output selectable
- Low pressure drop
- LED indication of no liquid & of flow rate





| Technical Specifications | | | |
|---|---|---|---|
| Max. flow L/min (Q4) | 8 | | |
| Transitional flow (Q2) | 0.4 | | |
| Min. flow L/min (Q1) | 0.1 | | |
| Output selectable | Pulse | Analogue | |

| Performance | |
|---|---|
| Accuracy for flow rates between Q2 and Q4 | 3% of reading |
| Accuracy for flow rates between Q1 and Q2 | 5% of reading |
| Resolution better than | 0.001L/min |
| Reverse flow | 0-8L/min |
| Response time | Better than 0.1s |

| Interface | | | |
|---|---|---|---|
| Connection 8 wires (RED supply +ve, BLACK ground, Blue NPN output, Yellow Data in, White PNP output, Orange Voltage output, Brown Current output, Green Ground connected internally to black) | | | |
| Supply | 8 - 24VDC (input current <15mA @ 24VDC) | | |
| Output selectable | 1000 pulses/L | 0-5Vdc | 4-20mA |

| Operation | |
|---|---|
| Principle | Ultrasonic transit time in-line flow cell |
| Temp. range (fluid) | -10°C to 85°C |
| Continuous fluid sound | Maintains performance regardless of fluid type, temperature or viscosity for speed measurement fluids with sound speeds 1250 - 1750 m/s |

| Physical characteristics | |
|---|---|
| Flow tube material | Glass filled plastic, Grivory HT1V-4FWA Black 9225 (FDA and EU approved for foodstuffs) |
| Flow tube internal diameter | 7mm |
| Connection thread | 3/8" straight tube |
| Internal bore of connection | 7mm |
| Suitable Pushfit adaptor (to fit 3/8" OD Tube) | John Guest Speedfit PI0412S |
| Maximum pressure | 10 bar |
| Case material | ABS black, Polylac PA-757 |
| Case integrity | Ultrasonically welded, not liquid proof |
| Connection | 8 core, PVC sheathed, 100cm long standard |
| Environmental protection | IP66 |

Ordering Code

UF 08 B 100

- Series
- Flow range (8= 0 to 8L/min)
- Cable length (cms)
- Output (B = selectable)

Cynergy3 Components Ltd.
7 Cobham Road
Ferndown Industrial Estate
Wimborne, Dorset BH21 7PE
Telephone +44 (0) 1202 897969

Email:sales@cynergy3.com

ISO 9001 CERTIFIED
UF08B 2016

www.cynergy3.com

## 5.4. Sensor de nivel

E+direct

Liquicap T FMI21

Sonda capacitiva para la medida de nivel continua en líquidos

# Liquicap T FMI21


Para más información:
www.e-direct.endress.com/fmi21

- No necesita calibración
- Materiales no corrosivos (fibra de carbono, acero inoxidable)
- Sonda diseñada para un funcionamiento seguro independientemente de la geometría del depósito

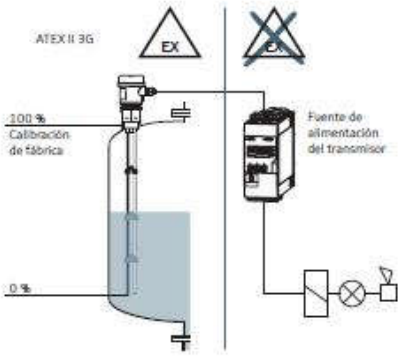
**Especificaciones generales:**

- **Producto:** conductividad del líquido ≥30 µS/cm
- **Longitud de la sonda:** 150 a 2500 mm
- **Presión de proceso:** -1...+10 bar
- **Certificaciones:** ATEX II 3G EEx nA IIC T6
- **Temperatura de proceso:** -40...+100 °C
- **Viscosidad:** máx. 2000 cSt

**Aplicaciones** El sensor Liquicap T FMI21 se emplea para medición continua de niveles en líquidos conductivos. Liquicap T FMI21 es especialmente apto para medir rangos de medida reducidos, por ejemplo en cisternas con líquidos agresivos tales como ácidos y bases. Funcionamiento seguro, independiente del material del depósito (plástico, acero inoxidable, hormigón...) o de su forma. Resistente a líquidos agresivos como ácidos o alkalis.

**Funcionamiento** La sonda, el medio y la varilla de toma de tierra (contrelectrodo) forman un condensador eléctrico. Cuando la sonda se halla al descubierto, detecta el valor de capacidad inicial. Al llenarse el depósito, el líquido va cubriendo la sonda y la capacidad del condensador aumenta. La electrónica de la sonda convierte ese valor medido de la capacidad, que es proporcional al nivel de líquido, en una corriente eléctrica dentro de un campo de valores de 4 a 20 mA, lo cual permite interpretar el nivel.

**Ejemplos de aplicación**




El sistema de medición consiste en:

- los componentes de una sonda capacitiva Liquicap T FMI21 con
- Electrónica FEI20
- Indicador y cubierta del cabezal (opcional)
- Una unidad de alimentación para el transmisor: RN221N, RMA42, RTA421 o RIA45/46

Endress+Hauser S.A.
C/Danubi, 12
08174 Sant Cugat del Vallés
España
Tel.: +34 934 803 366
Fax: +34 934 733 839
E-mail: info@es.endress.com

Endress+Hauser
People for Process Automation

## Datos técnicos

| **Entradas** | |
|---|---|
| Viscosidad máxima | 2000 cSt |
| Campo de medida | 0...2000 pF |
| Longitud de la sonda | 150...2500 mm |
| Span tolerado | ΔC = 10 pF...2000 pF |
| Frecuencia de medición | 250 kHz |
| Señal de entrada | Sondas sumergidas => capacidades altas<br>Sondas al aire => capacidades bajas |

| **Salida** (electrónica FEI20/4...20mA) | |
|---|---|
| Señal de salida | 3,8...20,5 mA |
| Corriente de activación | máx. 20 mA (<500 ms) |
| Señal de alarma | >21 mA |

| **Fuente de alimentación** | |
|---|---|
| Tensión para la conex. | U = 10...30 V CC, protección contra inversión de polaridad |
| Consumo | P <0,7 W |
| Consumo de corriente | I <22 mA |
| Entradas para cable | M20 × 1,5 (conexiones a rosca) |

| **Características de ejecución** (con la electrónica inserta instalada) | |
|---|---|
| Condiciones de funcionamiento de referencia | Temp. ambiente 23 °C, presión atmosférica, instalación vertical de la sonda desde arriba |
| Error de medición máx. | ≤1 % del valor de fondo de escala |
| Repetibilidad | 0,25 % del valor de fondo de escala |
| Tiempo de reacción | <2 s |
| Influencia de la temp. ambiente | <0,01 %/K (-40...+70 °C) longitud de la sonda 1 m |
| Tiempo de respuesta | 1 s |
| Calibración de fábrica | Una vez instalado, un recalibrado sólo es necesario si:<br>hay que ajustar los valores de 0 % y 100 % a las especificaciones particulares del cliente<br>en caso de haber reducido la longitud de las varillas de la sonda |

| **Condiciones de aplicación** | |
|---|---|
| Temp. ambiente | -40...+70 °C |
| Temp. almacenamiento | -40...+80 °C |
| Clase climática | Apto para el clima típico de los trópicos según DIN IEC 68 Parte 2-38 |
| Grado de protección | IP 66 |
| Resistencia a golpes | DIN EN 60068-2-27/IEC 68-2-27: 30g |
| Resist. a vibraciones | DIN EN 60068-2-64/IEC 68-2-64:<br>20...2000 Hz, 1 $(m/s^2)^2/Hz$ (con longitud mín. para las varillas de 150 mm) |
| CEM | Emisión de interferencias según EN 61326, clase B para equipos eléctricos; Inmunidad ante interferencias según EN 61326, Anexo A (equipos industriales) |
| Conductividad del medio | ≥30 µS/cm |
| Presión de proceso | -1...+10 bar |
| Temp. de proceso | -40...+100 °C |
| Resistencia a cargas laterales | 2 Nm |

| **Materiales en contacto con el medio** | |
|---|---|
| Varillas de la sonda | Varilla: 1.4404/316L; Opcional: fibra de carbono CFC; Junta: EPDM; Aislante: PP; Separador: PP |
| Conexiones a proceso | G 1½ A (PPS, DIN ISO 228/1) |
| Juntas | Junta anular para una conexión a proceso G 1½ A: Elastómero exento de fibra de amianto (a prueba de aceites, disolventes, vapor, ácidos y bases débiles) |

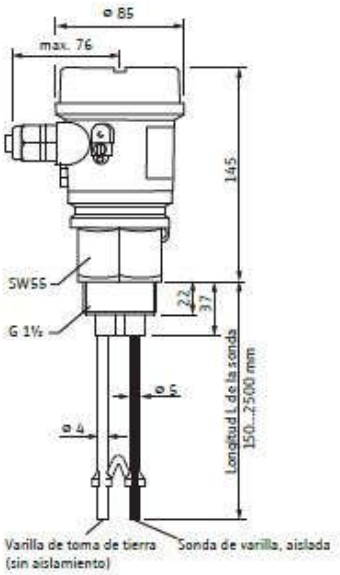
| **Indicador** | |
|---|---|
| LED verde | modo operativo (parpadeo lento), modo de calibración (parpadeo rápido) |
| LED rojo | para validación desde teclado (parpadeo corto), alarma o aviso (destello) |
| Indic. de valor medido en % | opcional |

| **Certificaciones** | |
|---|---|
| Certificación WHG | protección contra rebose según § 19 WHG (Alemania) |
| Certificación Ex | ATEX II 3G EEx nA IIC T6 |

E direct

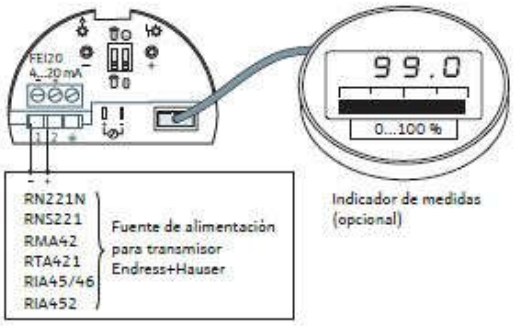
Liquicap T FMI21

## Dimensiones (en mm)




**Instalación conforme al manual de instrucciones.**

## Conexión eléctrica




## Tabla de precios

| Liquicap T FMI21 | | | | Referencia | Precio/unidad en € | | |
|---|---|---|---|---|---|---|---|
| Zona | Varillas de la sonda | Indicador | Longitud (mm)* | | 1 a 3 | 4 a 10 | 11 a 35 |
| No-Ex | 316L, L = 150...2500 mm | Sin | | FMI21-A1A1B1 | 336,– | 313,– | 296,– |
| | | Con | | FMI21-A1A1C1 | 374,– | 348,– | 329,– |
| | Fibra de carbono, L = 150...1000 mm | Sin | | FMI21-A1B1B1 | 361,– | 336,– | 318,– |
| | | Con | | FMI21-A1B1C1 | 399,– | 371,– | 351,– |
| | Fibra de carbono, L = 1 000...2500 mm | Sin | | FMI21-A1C1B1 | 387,– | 360,– | 340,– |
| | | Con | | FMI21-A1C1C1 | 424,– | 395,– | 373,– |
| No-Ex, WHG | 316L, L = 150...2500 mm | Sin | | FMI21-B1A1B1 | 356,– | 331,– | 313,– |
| | | Con | | FMI21-B1A1C1 | 394,– | 366,– | 346,– |
| | Fibra de carbono, L = 150...1000 mm | Sin | | FMI21-B1B1B1 | 381,– | 355,– | 336,– |
| | | Con | | FMI21-B1B1C1 | 419,– | 390,– | 369,– |
| | Fibra de carbono, L = 1 000...2500 mm | Sin | | FMI21-B1C1B1 | 407,– | 378,– | 358,– |
| | | Con | | FMI21-B1C1C1 | 444,– | 413,– | 391,– |
| ATEX II 3G EEx nA IIC T6, WHG | 316L, L = 150...2500 mm | Sin | | FMI21-C1A1B1 | 375,– | 349,– | 330,– |
| | | Con | | FMI21-C1A1C1 | 413,– | 384,– | 363,– |
| | Fibra de carbono, L = 150...1000 mm | Sin | | FMI21-C1B1B1 | 401,– | 373,– | 353,– |
| | | Con | | FMI21-C1B1C1 | 438,– | 408,– | 386,– |
| | Fibra de carbono, L = 1 000...2500 mm | Sin | | FMI21-C1C1B1 | 426,– | 396,– | 375,– |
| | | Con | | FMI21-C1C1C1 | 464,– | 431,– | 408,– |

* Por favor, especifique la longitud del sensor.

| Accesorios | Referencia | Precio/unidad en € |
|---|---|---|
| Tuerca de montaje G 1½" | 52014146 | 20,65 |
| Equipo de PP para reducción de la longitud de sondas | 52024300 | 10,57 |
| Indicador (pedir con la tapa transparente) | 52025604 | 100,89 |
| Tapa alta F16, transparente | 52025605 | 28,35 |

Precios aplicables en España hasta el 30.09.2017. Precios netos unitarios en €. Embalaje y transporte no incluidos. IVA no incluido.
Plazos de entrega: 48 horas o 5 días laborables – consultar en www.e-direct.endress.com los plazos de entrega exactos.

Para más información:
www.e-direct.endress.com/fmi21

Otros productos E-direct ...

Transductor de presión Cerabar T PMC131

Sonda de temperatura Easytemp TMR31

Transmisor de proceso RMA42

TD00068/23/ES/02.16

## 5.5. Amplificador

UNITED AUTOMATION

| DC MOTOR CONTROLLER MODULE | DCM X10335 |
|---|---|

**INTRODUCTION**

This general purpose, modulated, pulse-width, low voltage dc controller, can be operated in any of the following modes:

**Motor Control:** High Frequency (RT/RT1 no link) speed control set by a 5kΩ potentiometer.

**Lighting/Heating Control:** Low frequency (RT/RT1 linked) output level set by a 5kΩ potentiometer as above.

**Temperature Control:** Thermistor connected across RT/RT1, with a temperature range of 5-130°C. Temperature set by a 5kΩ potentiometer.

**APPLICATIONS**

Include speed control of low voltage, high frequency, dc motors, low voltage lighting and medium frequency heaters.

**FEATURES**

RoHS Compliant

- Manual or signal control.
- Temperature control with optional sensor.
- 180 or 350Hz selectable frequency ranges.
- Short-circuit protection.
- 6 to 24V dc supply voltage range.

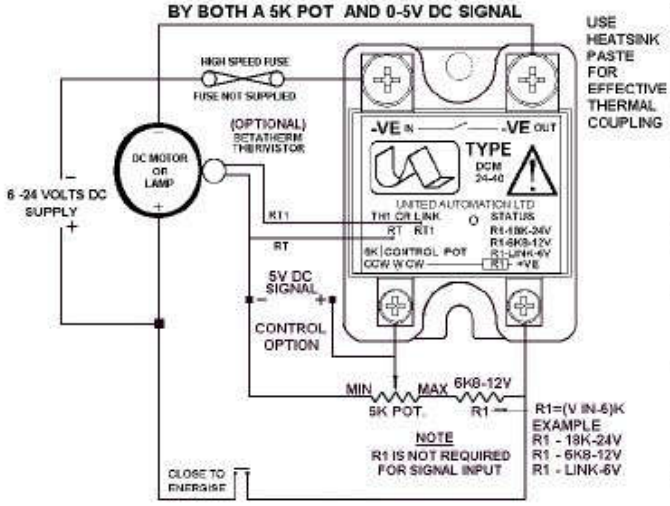
**INSTALLATION**

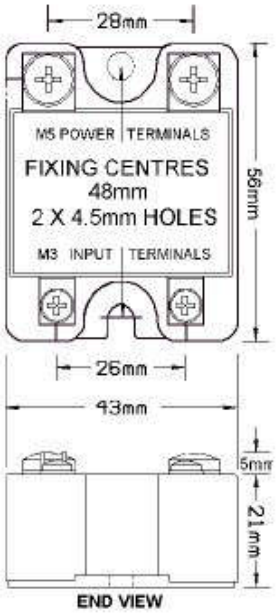
MOTOR CONTROL CONNECTIONS

**WARNING**

**SWITCH OFF SUPPLY BEFORE COMMENCING ANY SERVICE WORK.**

**ALL SET POINTS CAN BE CONTROLLED BY BOTH A 5K POT AND 0-5V DC SIGNAL**

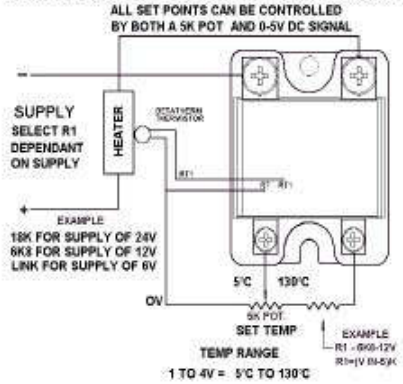


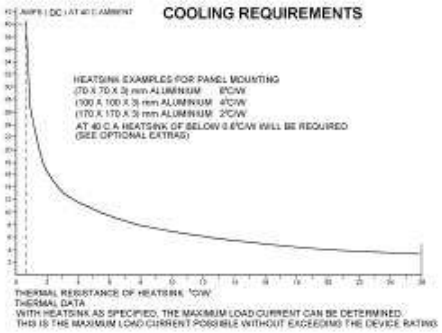



## INSTALLATION

TEMPERATURE CONTROL CONNECTION FOR 12V SUPPLY

WARNING
SWITCH OFF SUPPLY BEFORE COMMENCING ANY SERVICE WORK




## SPECIFICATIONS

| | |
|---|---|
| Maximum dc system line voltage | 24V dc |
| Unit limiting dc current | 40A dc |
| Control input voltage range | 0-5V dc |
| Control input current @ 5V typical | 1mA dc |
| High frequency mode (no link across RT and RT1) | 350Hz |
| Medium frequency mode (link RT and RT1) | 180Hz |
| Optional for temperature control (terminals RT & RT1): Thermistor type- Betatherm - 10K3A1 | 5 - 130°C |
| Unit operating temperature range | 0 to 65°C |
| Unit storage temperature range | 0 to 85°C |

## FUSING

It is recommended that semiconductor, fast-acting type fuses or circuit breakers (semiconductor - MCB) be used for unit/device protection. On initial operation some loads may need an increased factor of safety for unit/device protection (see SRA datasheet for further information).

## CE MARKING

This product family carries a CE marking. For information see recommendation section and contact our sales des. (see Declaration of Conformity).

## RECOMMENDATION

Other documents are available on request, which may be appropriate for your applications.

| CODE | IDENTITY | DESCRIPTION |
|---|---|---|
| X10229 | RFI | Filter recommendation: Addressing the EMC directive. |
| X10213 | ITA | Interaction: Uses for phase angle and for burst fire control. |
| X10255 | SRA | Safety requirements: addressing the Low Voltage Directive (LVD) including: thermal data/cooling, live parts warning, earth requirements and fusing recommendations. |
| P01.1 | COS | UAL conditions of sale. |

**NOTE:** It is recommended that installation and maintenance of this equipment should be done with reference to the current edition of the I.E.T. (formally I.E.E.) regulations (BS7671) by suitably qualified/trained personnel. The regulations contain important requirements regarding installation and safety of electrical equipment. Specific installers should refer to local and national regulations.

| | | |
|---|---|---|
| **ORDER CODE:** | State part number: DCM –24-40 | |
| Optional extras include:- | Betatherm 10K3A1 bead sensor only – | Stock code D80005 |
| | Betatherm 10K3A1 bead (type-X) sensor with 1m PTFE leads: | Stock code A26046 |
| | Betatherm 10K3A1 enclosed (type-E) sensor with 1m PTFE leads | Stock code A26036 |
| Further extras include:- | Heatsink assemblies for 40A capability; Heat sink paste; 5K potentiometer. | |

CE

**UNITED AUTOMATION LIMITED**

Southport Business Park
Wight Moss Way
Southport, PR8 4HQ
ENGLAND

Tel: 0044 (0) 1704 – 516500
Fax: 0044 (0) 1704 – 516501
enquiries@united-automation.com
www.united-automation.com

Page No. 2 of 2 Issue 5 Date 31/07/12

nqa. ISO 9001 Registered UKAS QUALITY MANAGEMENT 015

## 5.6. Bomba sumergible

**DATOS TÉCNICOS:**

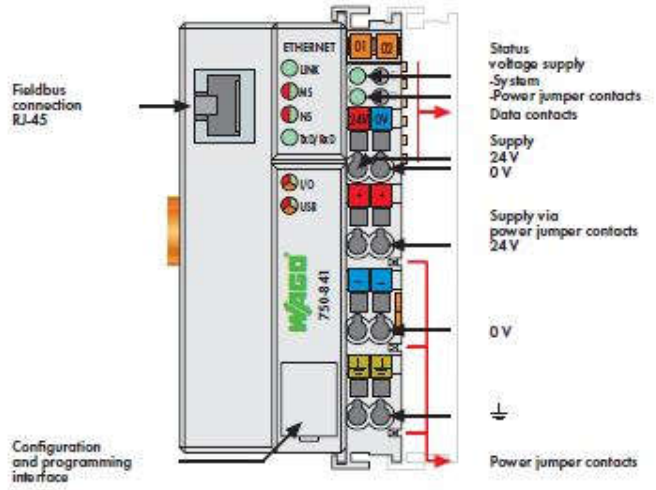
- Caudal máximo: 600 litros por hora (10l/min.) a desnivel cero.
- Altura máxima: 5,5 metros (caudal cero).
- Voltaje: 24V.
- Medidas: (Ø x Alt.) 38 mm x 104 mm
- Peso: 144 gramos.
- Cable de 1 metro. (polo positivo color marrón o negro)
- Admisión de corriente: 0,6 - 1,0 A.
- Temperatura máxima del agua: 60 ºC.
- Conexión de tubo para 10 mm ø.

# 5.7. PLC

## 750-841

### PLC - ETHERNET TCP/IP Programmable Fieldbus Controller

32-bit CPU, multitasking




This PLC connects ETHERNET to the WAGO-I/O-SYSTEM.

The controller automatically configures, creating a local process image which may include analog, digital or specialty modules. Analog and specialty module data is sent via words and/or bytes; digital data is sent bit by bit.

The IEC 61131-3 programmable controller is capable of 10/100 Mbit/s data rates, providing 512 KB program memory, 256 KB data memory and 24 KB retain memory. It has a battery-backed RTC and 32-bit multitasking CPU.

The PLC offers many different application protocols which can be used for data acquisition or control (MODBUS, ETHERNET/IP) or for system management and diagnostics (HTTP, BootP, DHCP, DNS, SNTP, FTP, SNMP and SMTP). For Web-based applications, HTML pages can be generated on an internal server. Programs are directly accessible via XML and ASP. Furthermore, the PLC incorporates library functions for e-mail, SOAP, ASP, IP configuration, ETHERNET sockets and file system.

| Description | Item No. | Pack. Unit |
|---|---|---|
| ETHERNET Controller 100 MBit | 750-841 | 1 |
| Product discontinuation | Last call: 28.02.2013 | |
| Product substitutes | 750-881, 750-880 | |
| ETHERNET Controller 100 MBit/s/T | 750-841/025-000 | 1 |
| Extended operating temperature range: -20 °C ... +60 °C | | |
| Product discontinuation | Last call: 28.02.2013 | |
| Product substitutes | 750-880/025-000 | |

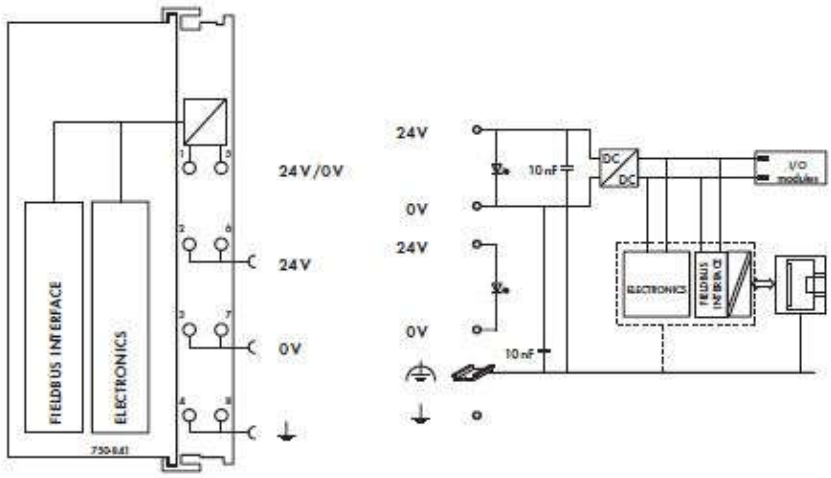
| Accessories | Item No. | Pack. Unit |
|---|---|---|
| WAGO-I/O-PRO V2.3, RS-232 kit | 759-333 | 1 |
| Miniature WSB Quick marking system | | |
| plain | 248-501 | 5 |
| with marking | siehe Seite 352 ... 353 | |

| Approvals | Also see "Approvals Overview" in Section 1 | |
|---|---|---|
| Conformity marking | CE | |
| Shipbuilding (versions upon request) | ABS, BV, DNV, GL, KR, LR, NKK, PRS, RINA | |
| UL 508 | | |
| ANSI/ISA 12.12.01 | Class I, Div. 2, Grp. ABCD, T4 | 750-841 |
| EN 60079-0, -11, -15 | I M2 Ex d I | 750-841* |
| EN 61241-0, -1, -11 | II 3 G Ex nA IIC T4 | 750-841* |
| | II 3 D Ex tD A22 IP6X T135°C | 750-841* |
| | * Permissible operating temperature: 0°C ... +60°C | |

| System Data | |
|---|---|
| No. of controllers connected to Master | limited by ETHERNET specification |
| Transmission medium | Twisted Pair S-UTP 100 Ω cat. 5 |
| Max. length of fieldbus segment | 100 m between hub station and 750-841; max. length of network limited by ETHERNET specification |
| Baud rate | 10/100 Mbit/s |
| Buscoupler connection | RJ-45 |
| Protocols | MODBUS/TCP (UDP), EtherNet/IP, HTTP, BootP, DHCP, DNS, SNTP, FTP, SNMP, SMTP |
| Programming | WAGO-I/O-PRO V2.3 |
| IEC 61131-3 | IL, LD, FBD, ST, FC |

WAGO Kontakttechnik GmbH & Co. KG Postfach 2880 - D-32385 Minden Tel.: +49(0)571/887-0 E-Mail: info@wago.com
Subject to design changes 16.10.2012 Hansastr. 27 - D-32423 Minden Fax: +49(0)571/887-169 www.wago.com

EtherNet/IP




| Technical Data | |
|---|---|
| Number of I/O modules | 64 |
| with bus extension | 250 |
| Fieldbus | |
| Max. input process image | 2 Kbytes |
| Max. output process image | 2 Kbytes |
| Max. input variables | 512 bytes |
| Max. output variables | 512 bytes |
| Configuration | via PC |
| Program memory | 512 Kbytes |
| Data memory | 256 Kbytes |
| Non-volatile memory (retain) | 24 Kbytes (16 Kbytes retain, 8 Kbytes flag) |
| Power supply | 24 V DC (-25 % ... +30 %) |
| Max. input current (24 V) | 500 mA |
| Efficiency of the power supply | 87 % |
| Internal current consumption (5 V) | 300 mA |
| Total current for I/O modules (5 V) | 1700 mA |
| Isolation | 500 V system/supply |
| Voltage via power jumper contacts | 24 V DC (-25 % ... +30 %) |
| Current via power jumper contacts (max.) | 10 A DC |

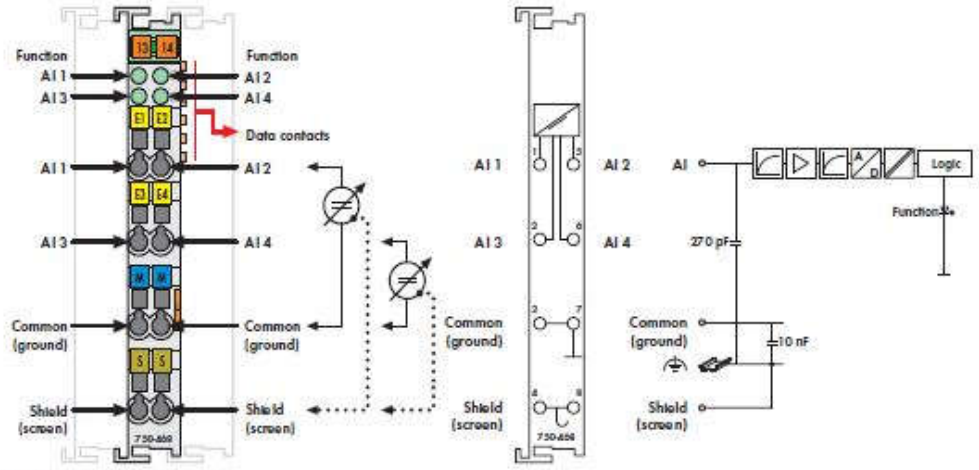
| General Specifications | |
|---|---|
| Operating temperature | 0 °C ... +55 °C |
| Wire connection | CAGE CLAMP® |
| Cross sections | 0.08 mm² ... 2.5 mm² / AWG 28 ... 14 |
| Stripped lengths | 8 ... 9 mm / 0.33 in |
| Dimensions (mm) W x H x L | 51 x 65 x 100 |
| | Height from upper-edge of DIN 35 rail |
| Weight | 184 g |
| Storage temperature | -25 °C ... +85 °C |
| Relative air humidity (no condensation) | 95 % |
| Vibration resistance | acc. to IEC 60068-2-6 |
| Shock resistance | acc. to IEC 60068-2-27 |
| Degree of protection | IP20 |
| EMC: CE - immunity to interference | acc. to EN 61000-6-2 (2005) |
| EMC: CE - emission of interference | acc. to EN 61000-6-4 (2007) |
| EMC: marine applications | |
| - immunity to interference | acc. to Germanischer Lloyd (2003) |
| EMC: marine applications | |
| - emission of interference | acc. to Germanischer Lloyd (2003) |

# 5.8. Módulos WAGO

## 750-468

### 4-Channel Analog Input Module 0-10 V

Single-ended (S.E.)




Delivered without miniature WSB markers

The analog input module receives signals with the standardized values of 0-10V.
The input signal is electrically isolated and will be transmitted with a resolution of 12 bits.
The internal system voltage supply is used for the power supply of the module.
The input channels of a module have one common ground potential.
The shield (screen) is directly connected to the DIN rail.

| Description | Item No. | Pack. Unit |
|---|---|---|
| 4AI 0-10V DC S.E. | 750-468 | 1 |
| 4AI 0-10V DC S.E. S5 [1] | 750-468/000-200 | 1 |
| 4AI 0-10V DC S.E./T | 750-468/025-000 | 1 |
| Extended temperature range: -20 °C ... +60 °C | | |

[1] Data format for S5 control with FB 251

| Accessories | Item No. | Pack. Unit |
|---|---|---|
| Miniature WSB Quick marking system | | |
| plain | 248-501 | 5 |
| with marking | see Section 11 | |

| Approvals | |
|---|---|
| Conformity marking | CE |
| Korea Certification | |
| Marine applications (versions upon request) | ABS, BV, DNV, GL, KR, LR, PRS, RINA |
| UL 508 | |
| ANSI/ISA 12.12.01 | Class I, Div. 2, Grp. ABCD, T4 |
| TÜV 07 ATEX 554086 X | I M2 Ex d I Mb,<br>II 3 G Ex nA IIC T4 Gc,<br>II 3 D Ex tc IIIC T135°C Dc |
| IECEx TUN 09.0001 X | Ex d I Mb,<br>Ex nA IIC T4 Gc,<br>Ex tc IIIC T135°C Dc |

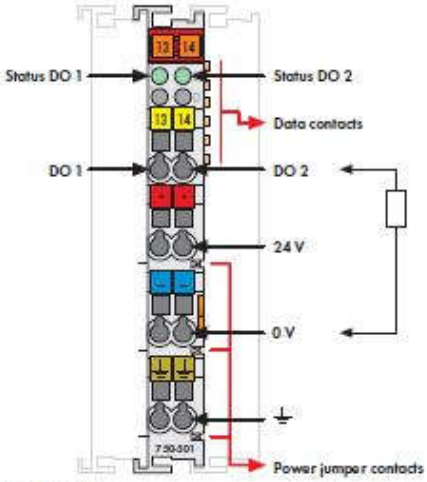
| Technical Data | |
|---|---|
| Number of inputs | 4 |
| Power supply | via system voltage DC/DC |
| Current consumption typ. (internal) | 60 mA |
| Input voltage (max.) | 35 V |
| Signal voltage | 0 ... 10 V |
| Internal resistance | 133 kΩ |
| Resolution | 12 bits |
| Conversion time (typ.) | 4 ms |
| Measuring error (25 °C) | ≤ ± 0.2 % of the full scale value |
| Temperature coefficient | < ± 0.01 %/K of the full scale value |
| Isolation | 500 V system/supply |
| Bit width | 4 x 16 bits data |
| | 4 x 8 bits control/status (optional) |
| Wire connection | CAGE CLAMP® |
| Cross section | 0.08 mm² ... 2.5 mm² / AWG 28 ... 14 |
| Strip lengths | 8 ... 9 mm / 0.33 in |
| Width | 12 mm |
| Weight | 52.5 g |
| EMC immunity of interference | acc. to EN 61000-6-2, marine applications |
| EMC emission of interference | acc. to EN 61000-6-4, marine applications |

WAGO Kontakttechnik GmbH & Co. KG
Subject to design changes 26.01.2016
Postfach 2880 - D-32385 Minden
Hansastr. 27 - D-32423 Minden
Tel.: +49(0)571/887-0
Fax: +49(0)571/887-169
E-Mail: info@wago.com
www.wago.com

# 750-501 / 753-501

## 2-Channel Digital Output Module 24 V DC

**Short-circuit protected; high-side switching**



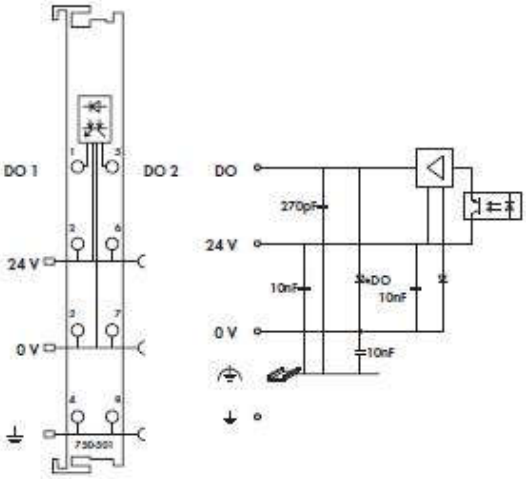




Fig. 750 Series
Delivered without miniature WSB markers

Control signals are transmitted from the automation device to connected actuators via the digital output module.

All outputs are short-circuit proof.

Field and system levels are electrically isolated.

| Description | Item No. | Pack. Unit |
|---|---|---|
| 2DO 24V DC 0.5A | 750-501 | 1 |
| 2DO 24V DC 0.5A/R* | 750-501/000-800 | 1 |
| * /R: Interference-free for safety function applications (see manual) | | |
| 2DO 24V DC 0.5A (without connector) | 753-501 | 1 |
| 2DO 24V DC 0.5A/R* (without connector) | 753-501/000-800 | 1 |

| Accessories | | Item No. | Pack. Unit |
|---|---|---|---|
| | 753 Series Connectors | 753-110 | 25 |
| | Coding elements | 753-150 | 100 |
| | Miniature WSB Quick marking system | | |
| | plain | 248-501 | 5 |
| | with marking | see Section 11 | |

| Approvals | |
|---|---|
| Conformity marking | CE |
| Korea Certification | |
| Marine applications (versions upon request) | ABS, BV, DNV, GL, KR, LR, NKK, PRS, RINA |
| UL 508 | |
| ANSI/ISA 12.12.01 | Class I, Div. 2, Grp. ABCD, T4 |
| TÜV 07 ATEX 554086 X | I M2 Ex d I Mb,<br>II 3 G Ex nA IIC T4 Gc,<br>II 3 D Ex tc IIIC T135°C Dc |
| IECEx TUN 09.0001 X | Ex d I Mb,<br>Ex nA IIC T4 Gc,<br>Ex tc IIIC T135°C Dc |

| Technical Data | |
|---|---|
| No. of outputs | 2 |
| Current consumption (internal) | 35 mA |
| Voltage via power jumper contacts | 24 V DC (-25 % ... +30 %) |
| Type of load | resistive, inductive, lamps |
| Max. switching frequency | 5 kHz |
| Output current (max.) | 0.5 A |
| Inductive load switch off energy dissipation W (max.) | 0.5 J; L max = 2 x W max / $I^2$ |
| Current consumption typ. (field side) | 15 mA / module + charge |
| Isolation | 500 V system/supply |
| Internal bit width | 2 bits |
| Wire connection | CAGE CLAMP® |
| Cross sections | 0.08 mm² ... 2.5 mm² / AWG 28 ... 14 |
| Strip lengths, 750/753 Series | 8 ... 9 mm / 0.33 in<br>9 ... 10 mm / 0.37 in |
| Width | 12 mm |
| Weight | 47.4 g |
| EMC immunity of interference | acc. to EN 61000-6-2, marine applications |
| EMC emission of interference | acc. to EN 61000-6-4, marine applications |

WAGO Kontakttechnik GmbH & Co. KG
Subject to design changes
26.01.2016
Postfach 2880 - D-32385 Minden
Hansastr. 27 - D-32423 Minden
Tel.: +49(0)571/887-0
Fax: +49(0)571/887-169
E-Mail: info@wago.com
www.wago.com

# 750-550, 750-556 / 753-550, 753-556

## 2-Channel Analog Output Module 0-10 V/±10V

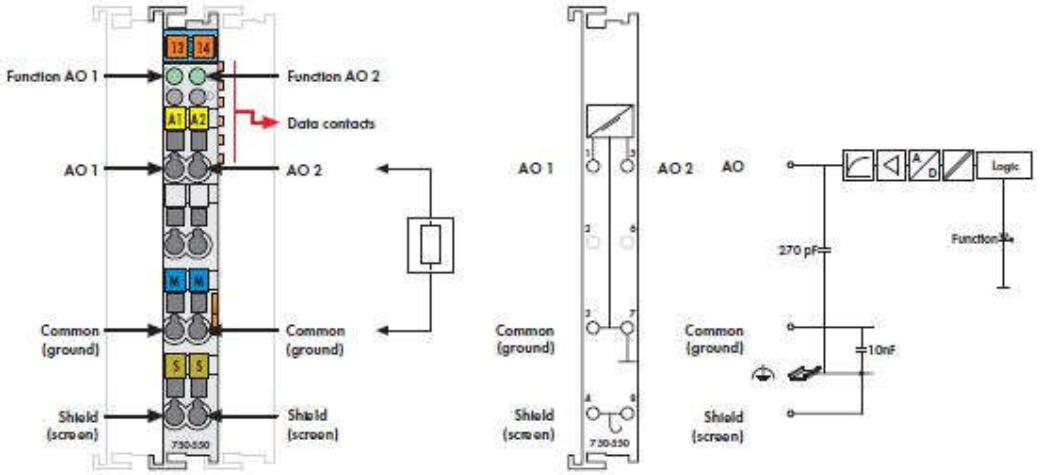



Fig. 750 Series
Delivered without miniature WSB markers

The analog output module creates a standardized signal of 0-10V or ±10V.

The output signal is electrically isolated and will be transmitted with a resolution of 12 bits.

Outputs are short circuit protected.

The internal system supply is used for the power supply of the module.

The output channels have one common ground potential.

| Description | Item No. | Pack. Unit |
|---|---|---|
| 2AO 0-10V DC | 750-550 | 1 |
| 2AO ± 10V DC | 750-556 | 1 |
| 2AO 0-10V DC/S5 [1] | 750-550/000-200 | 1 |
| 2AO ±10V DC/S5 [1] | 750-556/000-200 | 1 |
| 2AO 0-10V DC (without connector) | 753-550 | 1 |
| 2AO ±10V DC (without connector) | 753-556 | 1 |

[1] Data format for S5 control with FB 251

| Accessories | | Item No. | Pack. Unit |
|---|---|---|---|
|  | 753 Series Connectors | 753-110 | 25 |
| | Coding elements | 753-150 | 100 |
| | Miniature WSB Quick marking system | | |
| | plain | 248-501 | 5 |
| | with marking | see Section 11 | |

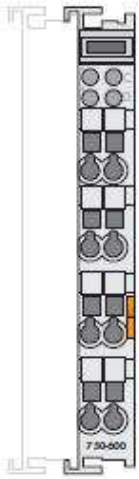
| Approvals | |
|---|---|
| Conformity marking | CE |
| Korea Certification | |
| Marine applications (versions upon request) | ABS, BV, DNV, GL, KR, LR, NKK, PRS, RINA |
| UL 508 | |
| ANSI/ISA 12.12.01 | Class I, Div. 2, Grp. ABCD, T4 |
| TÜV 07 ATEX 554086 X | I M2 Ex d I Mb,<br>II 3 G Ex nA IIC T4 Gc,<br>II 3 D Ex tc IIIC T135°C Dc |
| IECEx TUN 09.0001 X | Ex d I Mb,<br>Ex nA IIC T4 Gc,<br>Ex tc IIIC T135°C Dc |

| Technical Data | |
|---|---|
| No. of outputs | 2 |
| Current consumption (internal) | 65 mA |
| Power supply | via system voltage DC/DC |
| Signal voltage | 0 - 10V (750-550 / 753-550)<br>± 10V (750-556 / 753-556) |
| Load impedance | > 5 kΩ |
| Linearity | ±10 mV |
| Resolution | 12 bits |
| Conversion time | approx. 2 ms |
| Recovery time (typ.) | 300 µs |
| Measuring error (25 °C) | < ± 0.1 % of the full scale value |
| Temperature coefficient | < ± 0.01 % /K of the full scale value |
| Isolation | 500 V system/supply |
| Bit width | 2 x 16 bits data<br>2 x 8 bits control/status (option) |
| Wire connection | CAGE CLAMP® |
| Cross sections | 0.08 mm² ... 2.5 mm² / AWG 28 ... 14 |
| Strip lengths, 750/753 Series | 8 ... 9 mm / 0.33 in<br>9 ... 10 mm / 0.37 in |
| Width | 12 mm |
| Weight | 48.8 g |
| EMC immunity of interference | acc. to EN 61000-6-2, marine applications |
| EMC emission of interference | acc. to EN 61000-6-4, marine applications |

WAGO Kontakttechnik GmbH & Co. KG
Subject to design changes
26.01.2016
Postfach 2880 - D-32385 Minden
Hansastr. 27 - D-32423 Minden
Tel.: +49(0)571/887-0
Fax: +49(0)571/887-169
E-Mail: info@wago.com
www.wago.com

# 750-600

## End Module



**After the fieldbus node is assembled with the correct buscoupler and selected I/O modules, the end module is snapped onto the assembly.**

It completes the internal data circuit and ensures correct data flow.

| Description | Item No. | Pack. Unit |
|---|---|---|
| End Module | 750-600 | 1 |
| End Module/T | 750-600/025-000 | 1 |
| Extended temperature range: -20 °C … +60 °C | | |

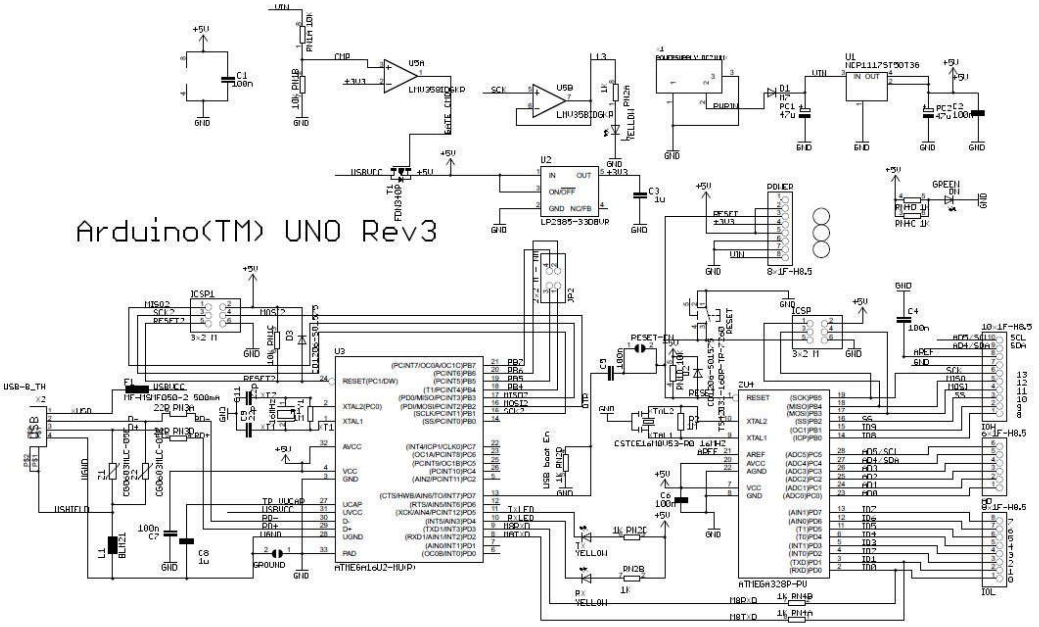
| Accessories | | Item No. | Pack. Unit |
|---|---|---|---|
| Miniature WSB Quick marking system | | | |
| | plain | 248-501 | 5 |
| | with marking | see Section 11 | |

| Approvals | |
|---|---|
| Conformity marking | CE |
| Korea Certification | |
| Marine applications | ABS, BV, DNV, GL, KR, LR, NKK, PRS, RINA |
| UL 508 | |
| ANSI/ISA 12.12.01 | Class I, Div. 2, Grp. ABCD, T4 |
| TÜV 07 ATEX 554086 X | I M2 Ex d I Mb,<br>II 3 G Ex nA IIC T4 Gc,<br>II 3 D Ex tc IIIC T135°C Dc |
| IECEx TUN 09.0001 X | Ex d I Mb,<br>Ex nA IIC T4 Gc,<br>Ex tc IIIC T135°C Dc |

| Technical Data | |
|---|---|
| Width | 12 mm |
| Weight | 46.4 g |
| EMC immunity of interference | acc. to EN 61000-6-2, marine applications |
| EMC emission of interference | acc. to EN 61000-6-4, marine applications |

WAGO Kontakttechnik GmbH & Co. KG
Subject to design changes
26.01.2016
Postfach 2880 · D-32385 Minden
Hansastr. 27 · D-32423 Minden
Tel.: +49(0)571/887-0
Fax: +49(0)571/887-169
E-Mail: info@wago.com
www.wago.com

## 5.9. Arduino UNO Rev3






## 5.10. ATmega328p