Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de una Tarjeta de Adquisición de Datos de Bajo Coste y su Aplicación a Prácticas de Laboratorio

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Oscar Bodoque Moreno

Tutor: Enrique Jorge Bernabeu Soler

2017-2018

Resumen

En el departamento de Ingeniería de Sistemas y Automática se usan tarjetas de adquisición de datos en cuatro de siete laboratorios. Puesto que se van quedando obsoletas respecto a los nuevos buses de datos (ISA, PCI) así como por los *drivers* para determinados programas y sistemas operativos, se pensó en una alternativa USB que, además de económica, tuviera mejor perspectiva de continuidad. Este proyecto aporta soluciones al problema. En los primeros capítulos con microcontroladores PIC (en adelante PIC) y su propia placa de circuito impreso y en los capítulos siguientes con la plataforma Arduino, cosa que permite ser implementada por el alumnado.

En este contexto una tarjeta de adquisición de datos (en adelante, TAD) es un componente hardware que se añade al ordenador para poder obtener datos analógicos del mundo real (como la velocidad, la temperatura, presión,...) procesar estos datos y posteriormente actuar sobre el sistema que se quiere controlar, también de manera analógica. La TAD realiza esta acción con convertidores digital – analógico y analógico – digital.

Palabras clave: tarjeta adquisición datos, PIC, Arduino.

Resum

Al departament d'Enginyeria de Sistemes i Automàtica es fan servir targetes d'adquisició de dades en quatre de set laboratoris. Ja que es van quedant obsoletes respecte als nous busos de dades (ISA, PCI) així com pels *drivers* per a determinats programes i sistemes operatius, es va pensar en una alternativa USB que, a més d'econòmica, tinga millor perspectiva de continuïtat. Aquest projecte aporta solucions al problema. En els primers capítols amb microcontroladors PIC (en endavant PIC) i la seva pròpia placa de circuit imprès i en els capítols següents amb la plataforma Arduino, cosa que permet ser implementada per l'alumnat.
En aquest context una targeta d'adquisició de dades (en endavant, TAD) és un component hardwarw que s'afegeix a l'ordinador per poder obtenir dades analògiques del món real (com la velocitat, la temperatura, pressió, ...) processar aquestes dades i posteriorment actuar sobre el sistema que es vol controlar, també de manera analògica. La TAD realitza aquesta acció amb convertidors digital - analògic i analògic - digital.

Paraules clau: targeta adquisició dades, PIC, Arduino.

Abstract

In the Department of Systems Engineering and Automation, data acquisition cards are used in four of seven laboratories. This cards are becoming obsolete with respect to the new data buses (ISA, PCI) as well as the drivers for certain programs and operating systems, we thought of a USB alternative that, in addition to being economical, would have a better continuity perspective. This project provides solutions to the problem. In the first chapters with PIC microcontrollers (hereinafter PIC) and its own printed circuit board and in the following chapters with the Arduino platform, which can be implemented by the students.

In this context, a data acquisition card (hereinafter, TAD) is a hardware component that is added to the computer to obtain real-world analog data (such as speed, temperature, pressure, ...) process this data and then act about the system that you want to control, also in an analogical way. The TAD performs this action with digital - analogue and analog - digital converters.

Keywords: data acquisition card, PIC, Arduino.

Tabla de contenidos

Índice de figuras

Índice de tablas

Agradecimientos

Agradezco al director del TFG, Enrique Bernabeu, su presión para realizarlo en fecha así como las sugerencias para poder hacerlo. Y a los profesores Raúl Simarro y Juan Manuel Herrero del departamento de Ingeniería de Sistemas sus explicaciones sobre los PID y su integración en LabView.

1 Introducción

Cada cierto tiempo las tarjetas de adquisición de datos (en adelante TAD) se van quedando obsoletas, bien por drivers de los sistemas operativos o bien por el hardware siempre cambiante de los ordenadores: ISA, PCI, PCI Express, etc.

Esto encarece mucho el mantenimiento de los laboratorios, a doce puestos cada uno y con un coste cercano a los quinientos euros si se trata de una TAD PCI y según el mercado. Para ahorrar costes y evitar en cierta medida el cambio de hardware temprano, se pensó en TAD USB. Estas tarjetas tienen un coste menor, cerca de doscientos cincuenta euros. Siendo más barato, sigue siendo mucho si hay que renovar todos los laboratorios, cosa que se hace permanentemente para que los PC soporten los nuevos sistemas operativos y versiones de programas, pero la idea del USB es buena.

Primero se diseñó una solución utilizando las TAD internas ISA de las que ya disponía el departamento. Para ello se usó un microcontrolador PIC (en adelante PIC) con puerto USB. Con este diseño el coste total fue de treinta y cinco euros por placa.

Dado que hasta las TAD con bus ISA se acaban y no es seguro que el resultado final quepa físicamente en los nuevos PC, el siguiente paso fue hacer una tarjeta independiente usando el mismo PIC pero incluyendo los convertidores D/A así como lo necesario para incluir al menos dos entradas y dos salidas analógicas. Con este diseño el coste se elevó por encima de los cincuenta euros.

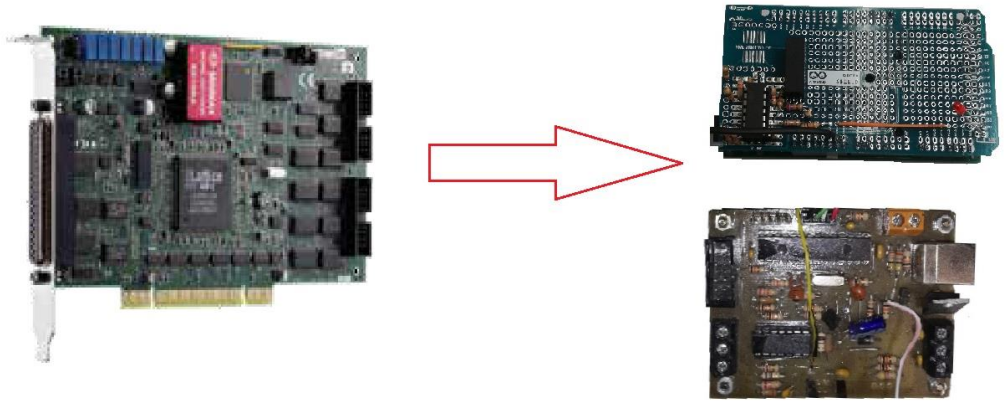
Para ambos diseños se programaron las rutinas y funciones suficientes para usarlas en entornos como MatLab y LabView y el soporte necesario a cualquier programa en el que ejecutar una rutina en C con acceso a un puerto serie normal. El desarrollo y programación de estos métodos se explica en el capítulo 2.

Aunque los diseños funcionaron se pensó en ampliarlos con una alternativa que pudiera construirse por los alumnos avanzados y con soporte futuro.

Esto llevó a pensar en Arduino, una plataforma barata y con soporte muy extendido, que en el caso del modelo Due lleva incorporado dos convertidores D/A de 12 bits. Para hacer el desarrollo barato se usaron en un principio las fuentes de alimentación del laboratorio, que en el caso del alumno podrían ser sustituidas por una fuente de PC rescatada. Posteriormente se pensó en añadir electrónica necesaria para alimentarlo desde el USB, al estilo de las placas comerciales, con un coste cercano a los cien euros. Estos diseños resultaron más caros que los anteriores con PIC, aunque al mismo tiempo, más al alcance de los alumnos, pues no tendrían que construir la placa de circuito impreso, y siempre inferior a lo que se pedía por una TAD USB comercial. Esto se cuenta en los capítulos 3 y 5.

Se hicieron pruebas de velocidad a todos los diseños y se probaron en casos reales de prácticas de laboratorio con procesos del departamento. Estas pruebas se muestran en los capítulo 5 y 6.

Ya en los capítulos 7 y 8 se avanza un poco más en un diseño, introduciendo el controlador o simulador de procesos diseñado, dentro de la misma placa, creando lo que se conoce como un sistema embebido. En el capítulo 8 se toman decisiones sobre el resultado final de todas ellas.



2 Primer acercamiento con PIC

En este capítulo se aborda el problema con un PIC de la casa Microchip serie 18F. Concretamente en PIC18F2550. En su interior cuenta con un interfaz USB completo [3] y es soportado por el compilador C de la propia casa, así como *drivers* para funcionar en la mayoría de sistemas operativos del mercado. De forma resumida diremos que es un microcontrolador con palabra de 8 bits capaz de funcionar con un reloj externo de 48 Mhz, las características completas del microcontrolador pueden consultarse en [3]. Para la realización de las placas de circuito impreso se usó Orcad Capture CIS [6] y Orcad Layout, disponibles en el software de la universidad.

2.1 Aprovechamiento de hardware ya disponible.

El primer acercamiento se produjo al proceder a la retirada definitiva del almacén de veinticinco TAD con bus ISA por no estar presente ese bus en los nuevos ordenadores. Dado que el bus ISA es sencillo de controlar, se pensó en aprovecharlas, así como todo el hardware asociado a ella (protecciones y conexionado). En el mercado existen adaptadores PCI-ISA, pero se descartaron por su elevado coste frente a comprar una TAD PCI nueva, por el paso atrás que supone y por la poca disponibilidad. La siguiente idea fue aprovechar los conocimientos del bus ISA y de microcontroladores para desarrollar una tarjeta que actúe como adaptador, pero desde USB.

La placa debe ser reducida pues debe ir en el interior del PC. Los componentes serán SMD y se elige un PIC con las patas justas para que sea reducido.

Lo primero es recopilar información de la TAD, concretamente la ACL 8112, para conocer sus registros, direcciones y función de cada uno. Esto se muestra en la Tabla 1 [1]. Aunque la tarjeta dispone de entradas y salidas digitales, contadores y generadores de interrupción, estos no son usados en las prácticas habituales, por lo que no se tienen en cuenta. Sólo se usarán los convertidores D/A y A/D.

Para conocer la dirección de cada registro se remite al manual de usuario. Con esta información y la función de cada uno de ellos, controlar la tarjeta como si estuviera conectada en un bus ISA original será sencillo.

El PIC será el encargado de controlar las líneas de dirección del bus ISA [2] para acceder a cada registro y a las líneas de datos para solicitar u obtener información, según se ve en el esquema de la Figura 1.

Tabla 1. Registros de la TAD ACL 8112

Dirección E/S	Lectura	Escritura
Base + 0	Contador 0	Contador 0
Base + 1	Contador 1	Contador 1
Base + 2	Contador 2	Contador 2
Base + 3	No usado	Control Contador 8254
Base + 4	Byte bajo A/D	Byte bajo Canal 1 D/A
Base + 5	Byte alto A/D	Byte alto Canal 1 D/A
Base + 6	Byte bajo Entradas Dig.	Byte bajo Canal 2 D/A
Base + 7	Byte alto Entradas Dig.	Byte alto Canal 2 D/A
Base + 8	No usado	Limpia Petición Interrup.
Base + 9	No usado	Control de Rango A/D
Base + 10	No usado	Canal Multiplexor
Base + 11	No usado	Control de modo
Base + 12	No usado	Disparo Software A/D
Base + 13	No usado	Byte bajo Salida Dig.
Base + 14	No usado	Byte alto Salida Dig.
Base + 15	No usado	No usado

El bus B del PIC se usa para intercambio de datos con el bus de la TAD, mientras que el bus A será el encargado de elegir las direcciones (y por tanto los registros), así como las señales de lectura y escritura (RD y WR respectivamente) que controlan la dirección en la que se leen o se escriben los datos.

El bus C se usa para varias cosas. RC0 se usa para seleccionar si el PIC entra en modo *bootloader* (este modo permite la reprogramación del PCI sin tener que desmontar el circuito, la reprogramación se hace desde el propio PC con bus USB). El bus USB es cosa de RC4 y RC5, que se conectan al PC a través de unas resistencias de protección, para evitar posibles problemas tanto en un lado como en otro. RC7 se usa para resetear la TAD desde el programa del PC.

RC1, RC2 y RC6 no se usan. RC3 en la versión USB es usado para filtrar la alimentación.

Algunas patas tienen doble función, como RB6 y RB7 que se usan para programar el PIC directamente mediante el conector ICSP. Para programarlo de esta manera se necesita un programador especial.

Con el jumper J2 podemos elegir si el circuito se alimenta desde el conector USB, útil durante el desarrollo, o desde la propia alimentación del PC. Esta alimentación entra por uno de los conectores disponibles para ello, J8 o J11. Estos conectores son los que se usan para alimentar los discos duros o las disqueteras en el interior del PC.

Por último, la señal AEN (*Adrress ENable,* Habilitar direcciones) [2] se pone a nivel bajo permanente, pues no hay conflictos posibles con otras tarjetas ISA.

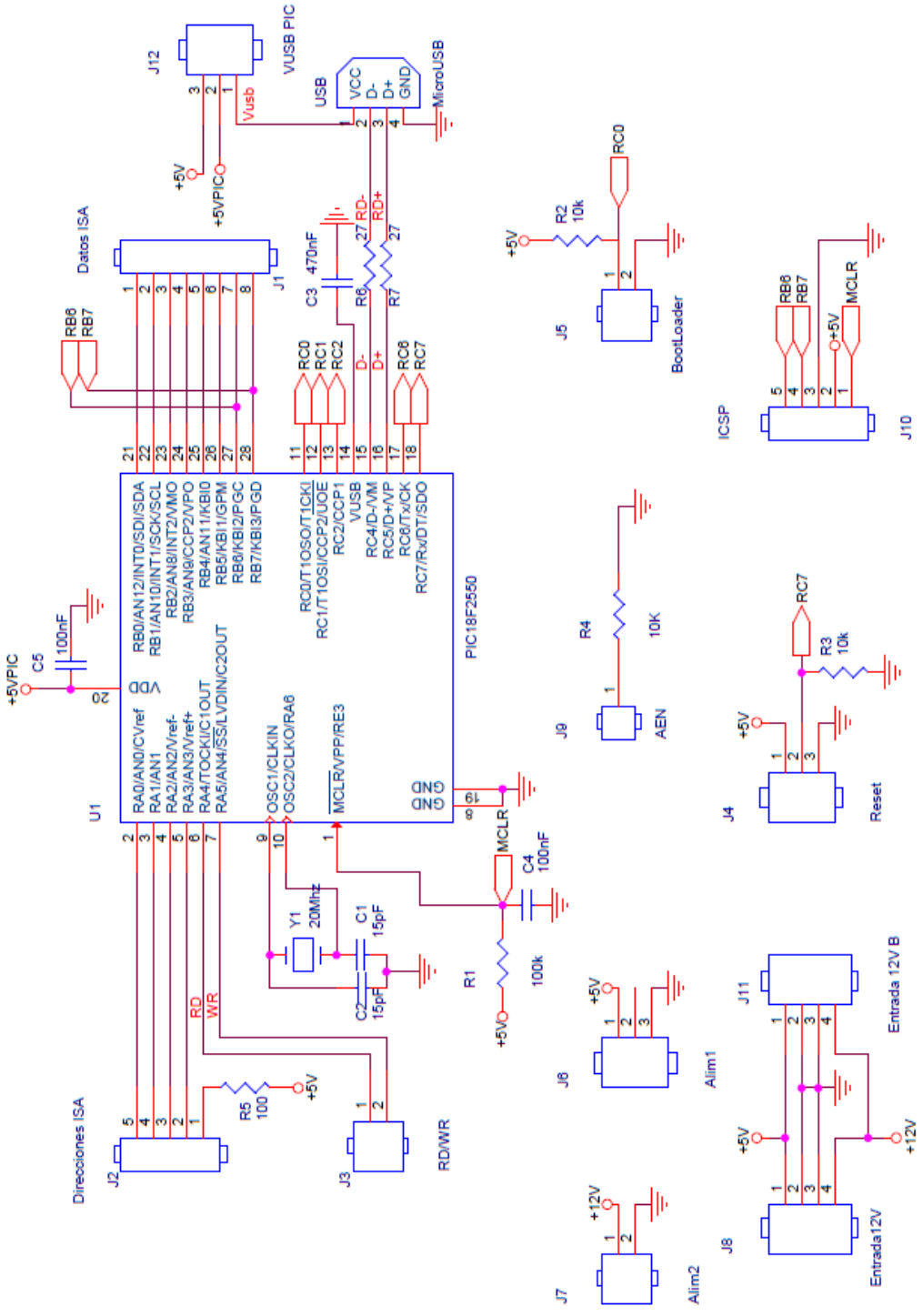


Figura 1. Esquema eléctrico PIC-ISA

En la toma de decisiones sobre si el acceso a la tarjeta sea por *polling* o de otra manera, se decide que sea por petición, para no saturar el bus USB y ralentizar el sistema sin motivo. En este contexto la petición consiste en que el programa que quiera acceder a la tarjeta realiza una petición, el PIC accede a la tarjeta ISA y responde a la petición, si toca.

Por ejemplo, en el caso de querer leer un valor analógico, el programa que se ejecuta en el PC "pide" la lectura por el canal correspondiente al PIC, el PIC a su vez hace la petición a la tarjeta, esta responde y el dato obtenido, lo manda al PC como respuesta. En el caso de escribir un valor, se manda la petición, el PIC la recibe y la pasa a la TAD, pero no "responde" nada al PC.

Este trasiego de datos se hace vía USB. El comportamiento será similar a un puerto serie. Esta decisión fue tomada por que prácticamente cualquier programa de adquisición de datos permite el acceso a esos puertos de origen sin drivers especiales, más allá del propio USB-serie que incorpora el SO. Otra razón para elegirlo: el puerto serie tiene soporte en los sistemas operativos habituales, Windows, Linux y Mac OS. En el departamento se comprueba que funciona correctamente con los programas Matlab, Simulink, LabView y Visual C bajo Windows.

El compilador que se usa es el MPLAB® C18 [4], suministrado gratuitamente por Microchip. Pese a la reducida potencia del microcontrolador elegido, frente a los nuevos diseños, la tasa de lectura y escritura máxima obtenida viene limitada por el propio bus USB, el programa usado y el S.O. Logrando accesos de 2 ms, más que suficiente para todas las prácticas del departamento.

Esta solución es la más barata de todas, a continuación un presupuesto posible de una distribuidor web nada barato. El primer prototipo de placa de circuito impreso se hizo en el propio departamento, pero posteriormente se encargó a un fabricante profesional.

Tabla 2. Coste TAD con PIC y tarjeta ISA

PIC18F2550-I/SO	4,15 €
10 Resistencias SMD 100 ohm	1,54 €
10 Condensador SMD 15 pF, 50 V	0,42 €
Cristal, 20 MHz, SMD	0,845 €
Conector USB, USB Tipo A	0,822 €
150 Resistencia SMD 220 ohm	1,53 €
Conector 5.08 mm, 4 Contactos, Macho	0,96 €
Conector recto, 2.54 mm, 40 Contactos	3,51 €
Conector acodado, 2.54 mm, 40 Contactos	3,94 €
Placa CI ya fabricada	11,2 €
Subtotal de productos:	28,92 €
IVA	6,07 €
Total (a falta de gastos de envío)	35,00 €

El bajo coste se debe principalmente al desarrollo y fabricación propias y a la reutilización de los convertidores AD y DA, situados en la placa ISA. En el siguiente desarrollo esto no es así y aumenta un poco. También se debe notar que varias compras sirven para más de una placa, solo que no es posible comprar menos. La diferencia en cualquier caso no es significativa respecto al coste.

2.2 Desarrollo final con PIC.

Ante la imposibilidad de introducir en los nuevos PCs del departamento el conjunto TAD-Adaptador, se da el siguiente paso: diseñar una TAD únicamente con el PIC y los circuitos necesarios para adaptar los distintos niveles. El PIC elegido posee internamente un convertidor AD de 10 bits capaz de tomar 100 k muestras por segundo [3], que a su vez se puede usar en 10 entradas. Teniendo en cuenta que solo pretendemos usar dos entradas, quedan 50 k muestras por segundo en cada entrada. Más que suficiente comparado con los dos milisegundos que podemos obtener vía USB (500 muestras por segundo). Para la salida analógica se usó en un principio las dos salidas PWM que también incluye el PIC, de 10 bits cada una. Sin embargo esta solución se desechó pues el rizado mínimo conseguido fue de 5 mV, demasiado para algunas prácticas. Para remediarlo se añadió un integrado el convertidor DA MCP4922 de 12 bits , controlado por SPI, que también está disponible en el propio PIC.

El esquema resultante de la segunda elección es el de la Figura 3.

U2 es el propio PIC, con su oscilador externo X1, su conector para la programación en la propia placa vía J8 (ICSP en este caso [3]) y la detección de USB conectado. U1 y U10 son operacionales cuádruples que con su circuitería asociada se encargan de adaptar las entradas bipolar de +/-10 V a la tensión soportada por el PIC, de 0 a 5 V de la siguiente manera. Se exponen solo las formulas simplificadas.

$$\text{Vou1A} = \left(Vi\ \frac{R2}{R1+R2}\right)\Delta u1 \qquad \text{Vou1D} = \left(Viu1D\ \frac{R4}{R6+R4} + 5\frac{R6}{R4+R6}\right)\Delta u10$$

Puesto que: R2= R1, R4= R6, Δu1 = 1, Δu10 = 1, y ViuD = Vou1A lo que llega a la pata del PIC es:

$$ViPIC = \frac{Vi}{4} + \frac{5}{2} = \frac{Vi}{4} + 2.5$$

esto se traduce en que cuando lleguen 10 V por la entrada, la tensión en el PIC será de:

$$Vi = 10 \rightarrow \quad ViPIC = \frac{10}{4} + 2.5 = 2{,}5 + 2{,}5 = 5\ V$$

Mientras que si la tensión es de -10 V

$$Vi = -10 \rightarrow \quad ViPIC = \frac{-10}{4} + 2.5 = -2{,}5 + 2{,}5 = 0\ V$$

Estos cálculos también se aplican a U1B y U1C y la otra entrada del PIC

U9 o U8 (se debe poner uno solo) son reguladores de precisión. Sirve para “centrar” la tensión, tanto de entrada como de salida. U3 es el convertidor DA de 12 bits controlado por SPI. U7 es otro operacional cuádruple que en este caso convierte los 0 a 5 V que salen del convertidor en +/-10 V de salida de la TAD.

Los cálculos necesarios y resumidos son los siguientes:

$$Vo = Viu1B\left(\left(\frac{R37+R38}{R39}\right)+1\right) - 2{,}5\left(\frac{R37+R38}{R39}\right) \quad Viu1B = VoPIC\Delta U1D\ \frac{R42+R40}{R41+R42+R40}$$

Sabemos que R37=R42= 18K, R8=R40=22K, R39=R41= 10K y ΔU1D= 1, nos queda que

$$Vo = \frac{4VPIC}{5}\left(\left(\frac{4}{1}\right) + 1\right) - 2.5\left(\frac{4}{1}\right) \rightarrow \quad Vo = 4VoPIC - 10$$

Así, la salida oscilará en:

$$VoPIC = 0\,V \rightarrow Vo = 4 * 0 - 10 = -10\,V$$

$$VoPIC = 5\,V \rightarrow Vo = 4 * 5 - 10 = 10\,V$$

R18 y R21 son la protección contra sobre corriente. El operacional tiene que asegurar que las tensión es similar en la dos entradas (- y +) así que entregará una tensión ligeramente superior a la salida, pero se igualará al pasar por R18 o R21.

Otra circuitería externa, ajena a la TAD propiamente dicha, son los transistores T1 y T2, y los componentes a su alrededor, que incluyen un conector DB9. Esta "extensión" es un puerto RS232 y se añadió para tener la posibilidad de usar la placa junto con algún PLC del departamento que no tiene entradas y salidas analógicas, pero sí con puerto serie. Funciona de manera similar a como lo hace con USB, solo con RS232. La velocidad de adquisición se ve muy reducida, pero no era crítica para esa aplicación.

Al calcular el coste de este diseño se comprueba que es muy similar al anterior, pues solo se añaden componentes de bajo coste al mismo tiempo que se eliminan otros que ya no son necesarios. Lo vemos en la Tabla **3**.

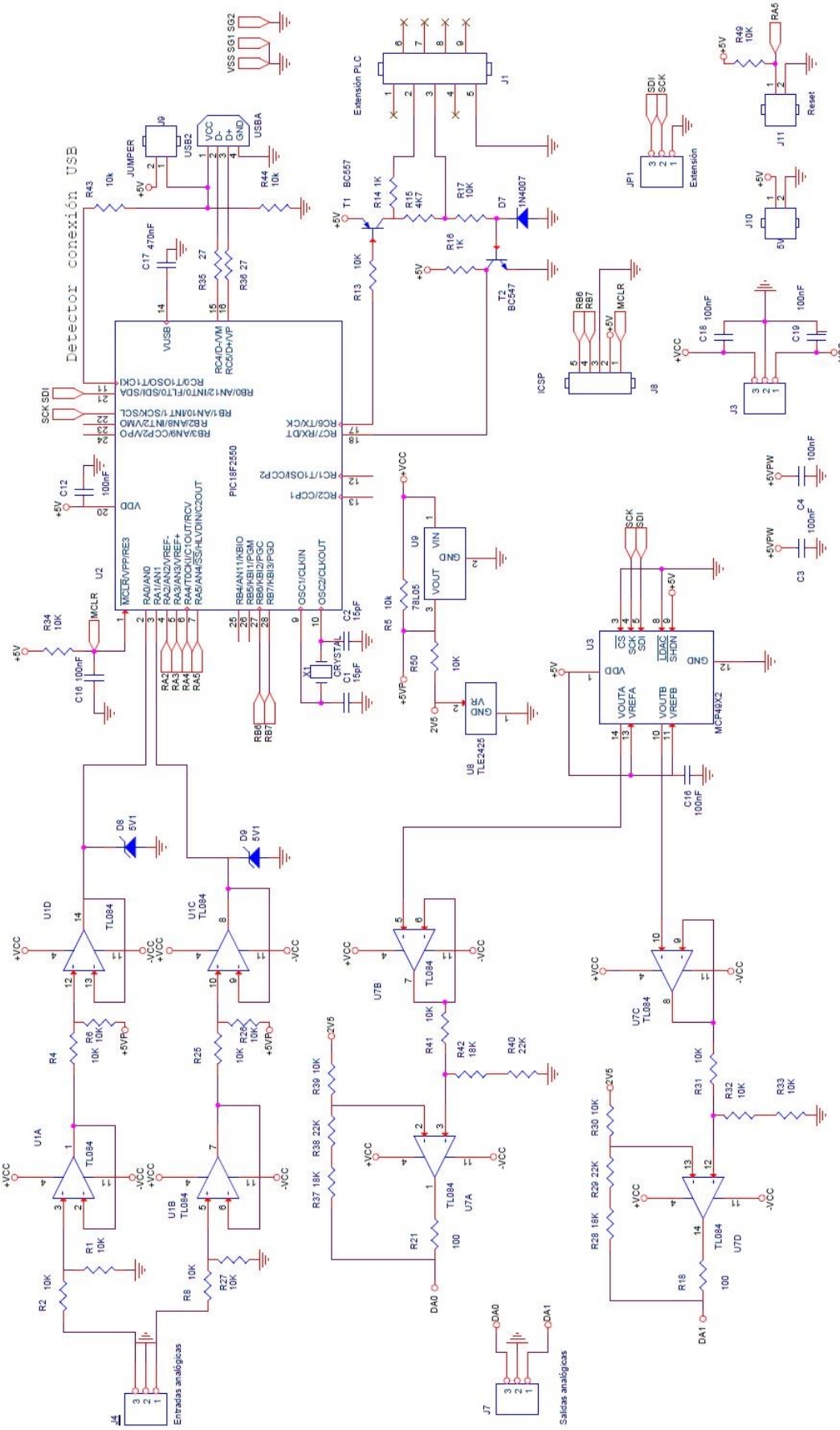


Figura 2. Tarjeta adquisición con PIC y convertidor DA.

Tabla 3. Coste de TAD con PIC y convertidor DA

PIC18F2550-I/SO	4,15 €
10 Resistencias SMD 100 ohm	1,54 €
10 Condensador SMD 15 pF, 50 V	0,42 €
Cristal, 20 MHz, SMD	0,845 €
Conector USB, USB Tipo B,	0,822 €
150 Resistencia SMD	1,53 €
3 Operacional cuádruple TL084	0,95 €
Convertidor DA doble 12 bits MCP4922	2,63 €
3 Bloque 3 terminales cable a placa	4,74 €
Placa CI fabricada (estimación)	15,5 €
Caja aluminio	9,70 €
Subtotal de productos:	42,83 €
IVA	8.99 €
Total (a falta de gastos de envío)	51.82 €

Se agrupan las resistencias en una misma línea, puesto que el pedido mínimo de todas ellas era de al menos cien y no se gastan cien en cada placa. Se añade el coste de la caja, pues esta solución se colocaría fuera del PC. Esto acarrea ciertos detalles como que se debe pensar en el acceso a las conexiones, con bananas de 4 mm si se quiere mantener el estilo empleado en el departamento. El coste de estas bananas y cables no está incluido, pues tampoco lo está en el caso de las TAD USB comerciales.

Tras las pruebas con esta circuitería, la mayoría de procesos era controlable sin problemas, pero la diferencia de precisión entre los datos obtenidos (de 10 bits) y los datos de control (12 bits) no gustó entre los usuarios finales. Además de la necesidad de alimentación externa, no presente en todos los laboratorios. Esto dio paso al diseño siguiente con Arduino.

3 Desarrollo con Arduino.

Con las ideas más claras de lo que se buscaba, 12 bits de entrada y otros 12 bits de salida, sin rizado por el PWM, se buscaron soluciones sencillas y asequibles (al menos en principio), dando un paso más y permitiendo en lo posible el desarrollo de la TAD por parte de algún alumno avanzado. La elección, por lo extendido de su uso, y la gratuidad en el software de desarrollo, fue la placa de desarrollo Arduino. Pero tuvo que ser el único que hasta el momento tiene tanto el convertidor AD como el DA de 12 bits, el Arduino Due [3].

Este microcontrolador sobrepasa en prestaciones al PIC anterior, por lo que se esperan obtener mejores especificaciones que con la anterior, en la práctica las diferencias no fueron significativas cuando trabajaba como TAD, pero sí cuando se integraba el control en el propio microprocesador. Las especificaciones completas de un Arduino Due podemos verlas en el este enlace.

Puesto que la tensión de salida de los convertidores DA es un poco especial, (entre 1/6 y 5/6 de la alimentación, que son 3.3 V) los valores de resistencias resultaron poco habituales para obtener los +/-10V necesarios. En vez de eso, se buscaron combinaciones de resistencias estándar con resultados igual de buenos. Durante el diseño se hizo usó del simulador LTSpice [5] de Analog Devices, poniendo especial atención en usar un operacional compatible con el que finalmente fue el elegido, un modesto TL084, barato pero de suficiente calidad.

Esta decisión fue tomada con la idea inicial de que un alumno avanzado pudiera construir el circuito con componentes de fácil adquisición, sin embargo se fue desechando poco a poco, pues el diseño dependía de las fuentes de alimentación del laboratorio, con tensión bipolar como la anterior.

De cara al departamento y a su uso masivo en prácticas, estas fuentes no están disponibles en todos ellos y el cableado puede ser delicado, pues un error en la polaridad supone la destrucción de los integrados operacionales. Para evitar esto en la medida de lo posible, se dotó al circuito de protección suplementaria. Pero finalmente y casi descartada del todo la construcción en casa debido a la cantidad de componentes, se optó por añadir un convertidor que proporcionara los +/-15V directamente desde el puerto USB. Con esto la tarjeta ya no dependía de la fuente y ya era más parecida a las USB tradicionales, tipo USB6001 de National Instruments [9], con un coste que ronda los doscientos cincuenta euros, pero a un menos de cien, y compatible con cualquier S.O., así como cualquier programa de control capaz de mandar y recibir datos por el puerto serie estándar (en realidad, un puerto serie desde USB).

El circuito es sencillo, pues lo importante está en la placa del Arduino Due, en la imagen solo se representa un canal de cada. Los operacionales U1A y U2A generan la salida de +/- 10V a partir de la salida del convertidor DA y uno (U1B) para adaptar esa tensión a la entrada del convertidor AD, que solo admite hasta 3.3V. La entrada entra invertida y será

necesario invertirla de nuevo digitalmente para poder enviarla al PC o al controlador programado de manera correcta. U2B se usa como amplificador de la tensión de referencia, un tanto especial y conseguida con el divisor de R23 y R14. Servirá para los dos canales de entrada.

Los cálculos para averiguar la tensión que llega al convertidor de Arduino son los siguientes:

$$\text{Vou1B} = \text{Vard} = Vref\left(\left(\frac{R2}{\frac{(R6+R7)*R27}{R6+R7+R27}}\right)+1\right) - \left(ViAD1\ \frac{R27}{R6+R7+R27} R\ 2/\frac{(R6+R7)*R27}{R6+R7+R27}\right)$$

$$Vref = 3{,}3\left(\frac{R14}{R13+R14}\right)\Delta U2B$$

Puesto que: R2= 10K, R6=39K, R7= 22K, ΔU2B = 1, R13= 27K y R14=8,2k.

$$Vard = 3{,}3\left(\frac{8{,}2}{35{,}2}\right)\left(\frac{610}{71}+1\right) - \left(\frac{10ViAD1}{71} * 1{,}164\right) = \ 1{,}665 - ViAD1 * 0{,}1665$$

Cuando lleguen 10 V por la entrada, la tensión en Arduino será de:

$$ViAD1 = 10 \rightarrow \quad Vard = 1{,}665 - 10 * 0{,}1665 = 0\ V$$

Mientras que si la tensión es de -10 V

$$ViAD1 = -10 \rightarrow \quad ViPIC = 1{,}665 - (-1{,}665) = 3{,}3\ V$$

La tensión de salida se obtiene de la siguiente manera:

$$VDA0 = \ -VoU1A\frac{R3}{R4//R8};\ \ VoU1A = 3V3\left(\frac{R12}{R9+R11+R12}+1\right) - VoDA0ard\frac{R1}{R5};$$

$$R4 = R8 = 22K;\ \ R4//\text{R8}= \frac{R4*R8}{R4+R8} = \frac{R4}{2} = 11K;\quad \text{R1=R5=100K; R12=10K, R9=18K, R11=12K}$$

$$VDA0 = \ -9{,}09VoU1A;\ \ VoU1A = 3V3(1{,}25) - VoDA0ard = 1{,}65 - VoDA0ard$$

$$VDA0 = 9{,}09VoDA0ard - \ 15$$

$$VoDA0ardmax = \frac{1}{6} * 3{,}3\ V = 0{,}66 \rightarrow VDA0 = 9{,}09 * 0{,}66 - 15 = 10V$$

$$VoDA0ardmin = \frac{5}{6} * 3{,}3\ V = 2.75 \rightarrow VDA0 = 9{,}09 * 2.75 - 15 = -10V$$

Los resultados obtenidos no fueron del todo los esperados, la alimentación USB era mala para el convertidor y se introducía mucho ruido en las señales tanto de entrada como de salida. Después de varios intentos infructuosos de reducir el ruedo con condensadores de filtro, la solución vino en aprovechar la propia capacidad del microprocesador e integrar un filtro digital interno.

Se hace notar que no se usan las utilidades que incorporan los programas Matlab o Labview para controlar la placa de Arduino directamente. De cara al programa de control, se trata de "una placa de adquisición de datos por puerto serie".

El coste de esta solución lo encontramos en la

Tabla **4**

Tabla 4. Coste TAD con Arduino Due

2 Terminales Estándar 3 Contactos, 5.08 mm	3,16 €
Placa desarrollo Arduino Due	29,41 €
2 Amplificador Operacional, Cuádruple, TL084ACN	2,06 €
Shield Arduino Due perforada	7,54 €
40 Resistencias varias	1,00 €
Convertidor DC/DC IR0515S	11,59 €
2 Tira 40 pines macho hembra	4,50 €
Caja aluminio 160x100x40mm	9,70 €
Cable USB A-Micro USB 1.8m	8,10 €
Subtotal de productos:	77,06 €
IVA	16,18 €
Total (a falta de gastos de envío)	93,24 €

Al igual que en el diseño anterior, se ha optado por añadir todas resistencias en una misma línea y la caja necesaria para que el diseño no quedara al descubierto. Las conexiones serían más sencillas gracias al cable USB y tampoco se cuentan las bananas necesarias.

Un problema asociado al redondear los cálculos para usar componentes de uso común y fáciles de localizar como resistencias con el estándar E12 y con un 5% de tolerancia en algunos casos o el operacional con offset "tan" elevado, es que el valor "0 V" casi nunca eran 0 v. A la hora de la verdad, este offset no supone un hándicap insuperable, pues se interpreta como parte del propio sistema a controlar, identificar o simular y puede usarse para que el alumno tenga que corregirlo.

Pero si el objetivo es construir una TAD lo más precisa posible, o los alumnos son todavía de los primeros cursos, mandar "un cero" y que el motor no se pare, es difícil de asumir: hubo que remediarlo. La solución vino de nuevo de aprovechar la parte digital. Se diseñó un programa en Labview que encontraba el valor de offset necesario para que anulara la desviación de la parte física analógica. Mediante una orden que se puede llamar "especial", el microcontrolador guarda internamente ese offset en la memoria EEPROM y así lo tiene disponible cada vez que se inicia.

La intervención del usuario es necesaria, pero solo una vez por tarjeta o en el futuro cuando envejezcan los componentes lo suficiente como para que este valor cambie de manera ostensible.

También esto fue un problema en el caso de Arduino, pues no posee memoria EEPROM interna. Dado que la escritura solo se hará unas pocas veces en la vida de la TAD, se pensó en usar la librería DueFlashStorage [12] para simularla. Esta librería usa la flash

de programa para guardar datos que permanecerán en memoria incluso con el procesador apagado. El trabajo con la memoria flash se debe hacer borrando y escribiendo bloques enteros y no permite muchas escrituras, estos problemas no lo son tanto si pensamos que una vez calculado y guardado el valor, no se vuelve a escribir en mucho tiempo.

Otra opción, dado que los valores de offset serán pequeños, es utilizar los bits libres que tiene el registro de configuración del procesador llamado GPNVM (*General Purpose Non-Volatile Memory,* Memoria no volátil de uso general). Los bits 3 a 8 no tienen uso conocido y dan para un valor de offset de +/- 78 mV. Sin embargo, en el modo simulador o control PID, no son suficientes y se ha de usar el otro método.

4 En el lado del PC

Los diseños anteriores se controlan a través del puerto serie virtual que se crea. Esto permite controlarlos desde cualquier S.O. y programa que permita el acceso a ese puerto. Normalmente todos. Nos centraremos en Windows, pues es el sistema que tienen todos los laboratorios de prácticas del departamento.

Los programas que usan las TAD son Labview, Matlab y en menor medida, Visual C. Se diseñaron los recursos necesarios para cada uno de ellos, .vi en el caso de Labview, .m en el caso de Matlab y .h en el caso de Visual C.

Cuando se usa la TAD ISA reciclada, el PIC contempla también la lectura y escritura de valores digitales en sus salidas, se diseñaron los .vi necesarios para ello, pero no los expondremos, pues son bastante similares a los mostrados y en el objetivo final no era imprescindible.

4.1 Escritura de un valor analógico

Pare hacer uso de la TAD en Labview se usan las opciones de comunicación serie incluidas en las librerías VISA [10]. Siguiendo la Figura **5** de izquierda a derecha, nos encontramos con la entrada del puerto a usar (se puede poner más de una TAD por ordenador) así como se sondea si hubo un error anterior. Cerramos el puerto antes de empezar cualquier operación con VISA C (*Clear*). Esto se hace pues al ser LabView multihilo se observó que otro hilo podía retener el puerto serie aunque ya hubiera acabado, dejando las demás operaciones en el aire, a la espera de que se liberara. Una vez liberado el puerto, se limpia el buffer y se manda la petición al PIC, se hace con VISA W (*Write*). La petición se realiza montando una cadena con "lo que se pide". Y la cadena se monta con los siguientes datos, dado que es una operación de escritura en un canal de salida, necesitamos el voltaje a entregar, la orden "salida" y el canal por el que queremos esa tensión. La orden "salida" se codificó con el número `5Xh`. En la X irá el número de canal a escribir, concretamente el 0 ó el 1, dado que solo hay dos canales de salida. Este valor viene dado por la entrada "Channel".

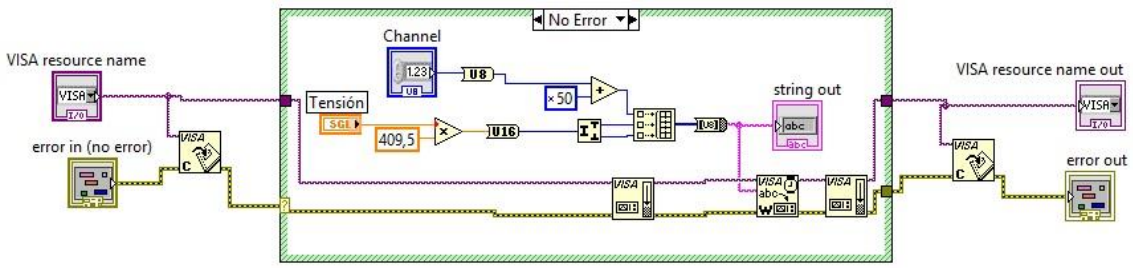


Figura 5. Escritura de un valor analógico de la TAD

Como se debe mandar un dato analógico por un puerto digital, se realiza la conversación a entero multiplicando por 409.5 en la solución que usa la TAD ISA. Este valor no es elegido al azar. Dado que los valores máximo y mínimo que esa tarjeta puede sacar son +10 V y 0 V respectivamente, al multiplicar por 409.5 nos quedan 4095, el mayor número entero disponible en 12 bits y 0, el menor. El valor resultante se convierte a un entero de 16 bits (aunque posteriormente solo se aprovechen 12). Dado que el puerto serie está configurado para mandar datos de 8 bits, se divide en dos y se monta la cadena que el PIC interpretara como petición.

En el ejemplo, Para mandar el valor 5 V por el canal 1 la cadena resultante queda: `510800h`. Cuando el PIC reciba esto, sabrá que es una operación de escritura en una salida analógica y efectuará las operaciones necesarias para que esto se lleve a cabo.

Cuando se trate de la TAD diseñada enteramente con componentes nuevos, la salida será de -10 V a + 10 V. En este caso la conversión es ligeramente diferente. Primero se sumará 10 al valor del voltaje, con lo que obtenemos un valor comprendido entre 0 y 20. Se multiplica entonces por 204.75, y el resultado es que los +10 V de entrada (20 en la entapa intermedia) se convertirán en 4095, de nuevo el mayor número posible con 12 bits, y cuando sea 0 V, será también 0. Posteriormente los operacionales situados después del convertidor se encargan de convertir de nuevo la tensión en +/-10 V. Como detalle, mandando la misma cadena que usábamos antes para escribir 5V obtendríamos ahora 0 V, también la mitad de la escala.

El valor que se muestra en la salida *string out* solo sirve para propósitos de depuración y es la cadena resultante a mandar.

Para Matlab [9][11] y Visual C++ la cosa es más sencilla, ya que se limita a usar código.

```
s1 = serial('COM17', 'BaudRate', 921600, 'Terminator', '', 'StopBits', 1);
fopen(s1);
%% Aquí se calcularía el dato en voltios a mandar
%% datovoltios = lo que sea;

if datovoltios<-10
    datovoltios=-10;
end
if datovoltios>10
    datovoltios=10;
end;

%convierto el datovoltios entre -10 y 10 en la salida entre 0 y 10
datovoltios=(datovoltios+10.0)/2.0;

datoentero=round(datovoltios*409.5);

bytealto=floor(datoentero/256);
bytebajo=datoentero-bytealto*256;

switch canal
    case 0
      mensaje=[char(hex2dec('50')),char(hex2dec('00')),char(bytealto),char(bytebajo),char(0)];
    case 1
      mensaje=[char(hex2dec('51')),char(bytealto),char(bytebajo)];
end;

fwrite(s1, mensaje, 'uchar');
fclose(s1);
```

4.2 Lectura de un valor analógico

Ya hemos comentado que la TAD diseñada con el PIC realiza las operaciones bajo una "petición". En el caso de la lectura de un valor analógico, el proceso a seguir es, realizar la petición de lectura por el canal deseado y esperar a que nos llegue la respuesta. Esta espera puede llegar a ser crítica si por algún motivo el PIC no respondiera, de ahí que se haya previsto un *timeout* para desechar la operación si eso se produce. El dato vendrá en formato digital y la idea es que la salida de la .vi sea el voltaje leído, es decir analógico. Luego hay que realizar la conversión aquí. Todo esto puede fallar en algún momento y para evitar problemas en el futuro, esta solución prevé posibles fallos. Es por ello que está .vi sea más complicada que la anterior.

Los primeros pasos sí son iguales, cerrar el puerto, borrar el buffer y montar la cadena necesaria para enviar la petición para enviarla con VISA W. Se elige el valor `AXh` para decirle al PIC que es lectura, en la X se enviará el canal a leer. Recordar que teníamos dos modelos de TAD con PIC, la que usa una TAD ISA y la que no incluye nada viejo. Cuando se use la TAD ISA, el valor de X estará entre 0 y 16, pues esos son los canales de entrada disponibles, mientras que cuando se use la otra, las salidas se limitarán a dos.

No hay nada más que montar, así que para leer el canal 3 la cadena a mandar sería simplemente `A3h`.

Una vez que se manda esta cadena, el PIC la interpreta como que se quiere leer el canal analógico 3 y se pone en marcha para realizar dicha lectura. El PC se queda esperando el valor que vendrá en forma de cuatro bytes, pero solo durante el tiempo indicado en el *timeout*. Si todo ha ido bien, pasamos al paso siguiente.

Aunque es cierto que un puerto serie virtual realizado con un USB es muy difícil que se partan las cadenas, el programa debe ser robusto y contemplar esa posibilidad, de ahí todo lo que viene a continuación, que no es más que asegurarse que los cuatro bytes leídos vienen en el orden que toca y que efectivamente son lo que estábamos esperando. La cadena que deberíamos leer estará compuesta por cuatro bytes, los dos primeros son de comprobación el primero debe ser igual al que mandamos nosotros como petición, en el ejemplo `A3h`, el siguiente es un valor de desplazamiento. No olvidemos que estamos leyendo valores analógicos, así que es posible leer un `A3h` que nos pueda confundir. Lo que es imposible (puesto que solo tenemos 12 bits) es que después venga un `AAh`, que es el valor que hemos elegido para este segundo byte. Si esto viene mal, y hemos leído cuatro bytes, se trata de un error. Está contemplado también, aunque es más redundante, que se hayan leído solo tres bytes por que salte el *timeout* antes de tiempo, entonces se lee uno más y se realizan las mismas comprobaciones. Todas las demás posibilidades se traducen en error. Otra razón para usar tanta comprobación de errores es el uso secundario de la tarjeta con el puerto serie RS232 junto con los autómatas.

Estando seguros que los datos leídos son buenos, pasamos a la conversión de datos digitales en valores analógicos. Esto se realiza de manera inversa a lo que vimos antes y el valor que se usa ahora para "añadir decimales" es 0.0048828125 que tampoco es elegido al azar. Se trata del resultado que se obtiene al dividir 20 entre 4095. 4095 será el valor máximo que puede leer la TAD, se corresponde con +10 V y al multiplicar por 0.0048828125 nos sale 20 V (en realidad 19.995 V) de ahí que luego se resten 10,

obteniendo de nuevo 10, los +10 V deseados. Si el valor leído fuera 0, el producto también sería 0 y al restar los 10, tendríamos los -10 V. Cubriendo todo el rango de lectura de la TAD.

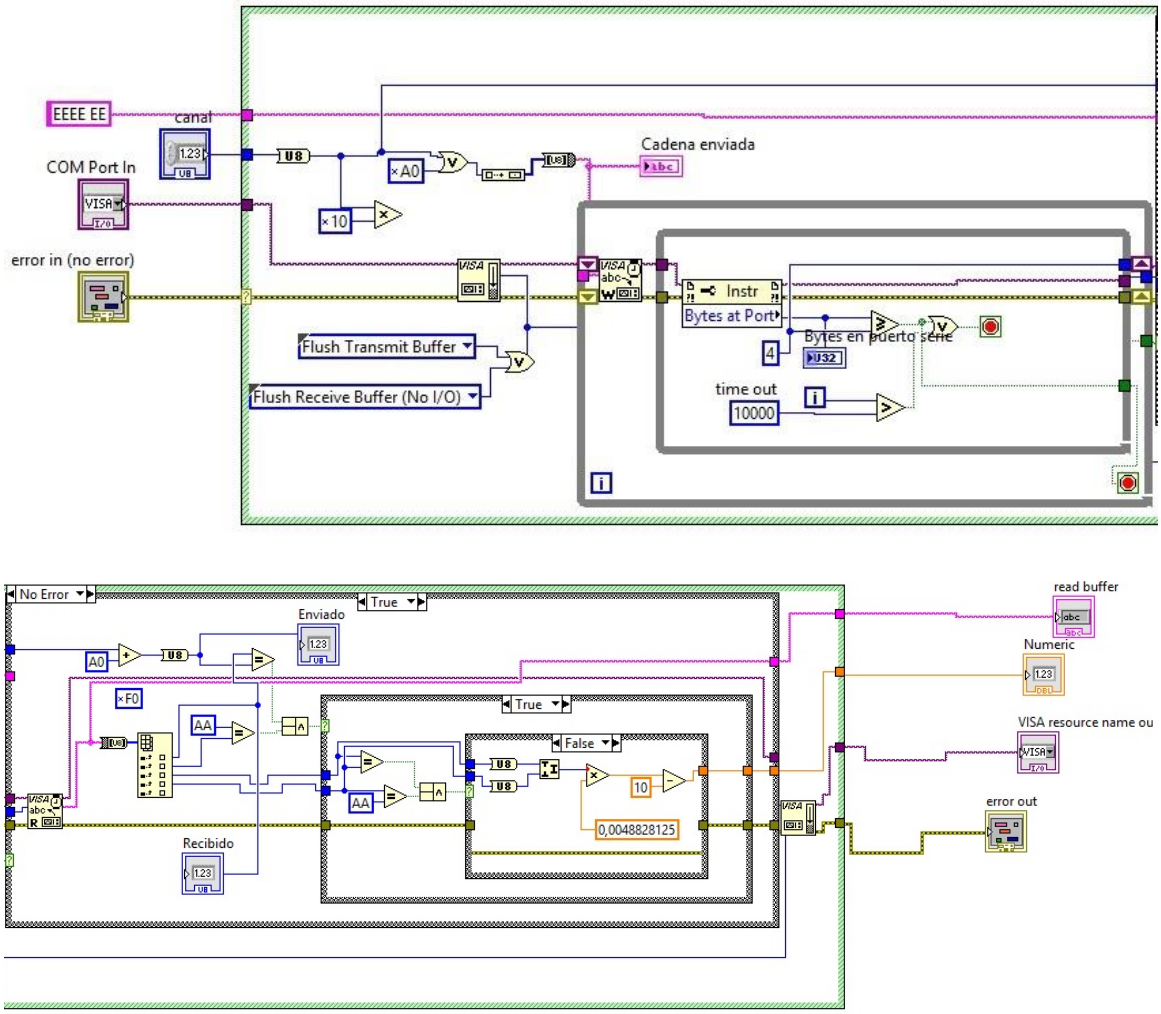


Figura 6. Lectura de un valor analógico en la TAD.

Las siguientes líneas leen el valor analógico del canal 0 en Matlab [9] [11]. Para Visual C sería muy parecido.

```
s1 = serial('COM17', 'BaudRate', 921600, 'Terminator', '', 'StopBits', 1);
fopen(s1);
%Petición del dato
fwrite(objetoserie, char(hex2dec('a0')), 'uchar');

%Lectura del dato pedido
dato = fread(objetoserie,4,'uchar');

%Conversión del dato a tensión +/-10
datoentero=dato(3)*256+dato(4);

datovoltios=(datoentero/204.8) - 10.0;
fclose(s1);
```

4.3 Cambiar configuración TAD por defecto

Se realizó otro .vi para configurar el arranque de la TAD. Se usó para configurar algunos parámetros como el uso de la TAD: simulador, controlador o TAD habitual. Otras características a configurar fueron: seleccionar si cada salida analógica sería bipolar o unipolar, importante si la TAD se usaba con ciertos procesos del departamento que solo admiten esas tensiones, programar el valor calculado para eliminar el offset de los componentes y algunos detalles propios del PIC como el arrancarlo en modo *download* (por usar la terminología actual usada con los móviles). Esto permite la actualización del firmware directamente por el puerto USB, sin tener que acceder al jumper disponible en la tarjeta y por tanto sin desmontar el lugar donde estuviera implantado, importante especialmente en la TAD se ubicaba en el interior del PC.

Este programa se realizó solo en Labview pues su uso es más intuitivo que cambiar cadenas en Matlab o Visual C.

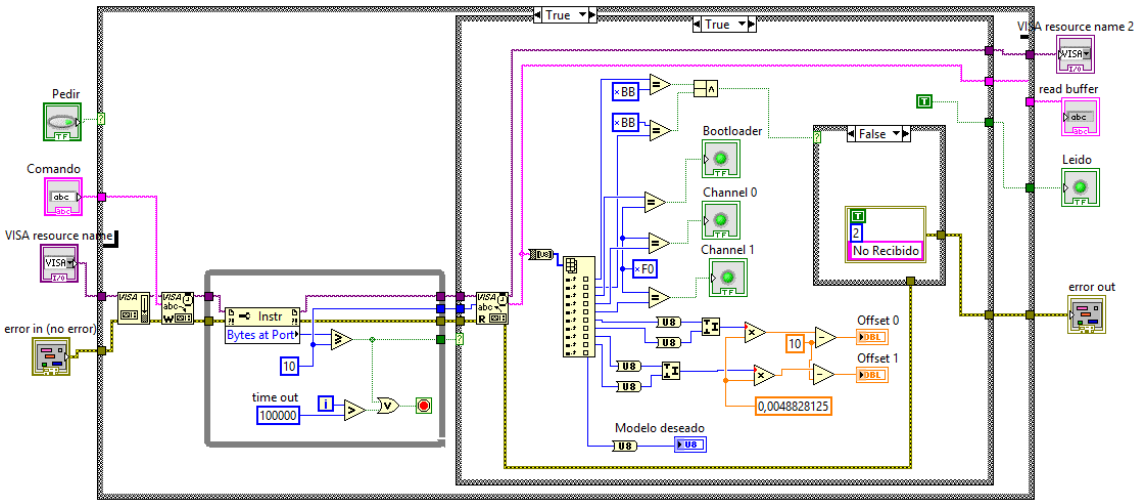


Figura 7. Solicitud de datos de setup guardados en la TAD.

Al igual que en las anteriores, se manda un comando a la TAD (`B0h`) y ella responde con la configuración. El mensaje correcto sería el que incluye dos veces seguida el byte `BBh`. De no ser así, se estima que la trama es incorrecta y se desecha. La información que viene a continuación es por este orden: si se ejecutará el bootloader en el próximo reinicio, si el canal 0 es bipolar, si lo es el canal 1 y los offset de los canales 0 y 1.

Nótese que lo que determina si es una cosa u otra es que el dato recibido sea `F0h`. La TAD debe mandar `F0h` si el bit está activo y `FFh` está inactivo. Cuando el modelo sea `00h`, la tarjeta se comportará normal, `01h` será un controlador y a partir de ahí, los distintos modelos programados.

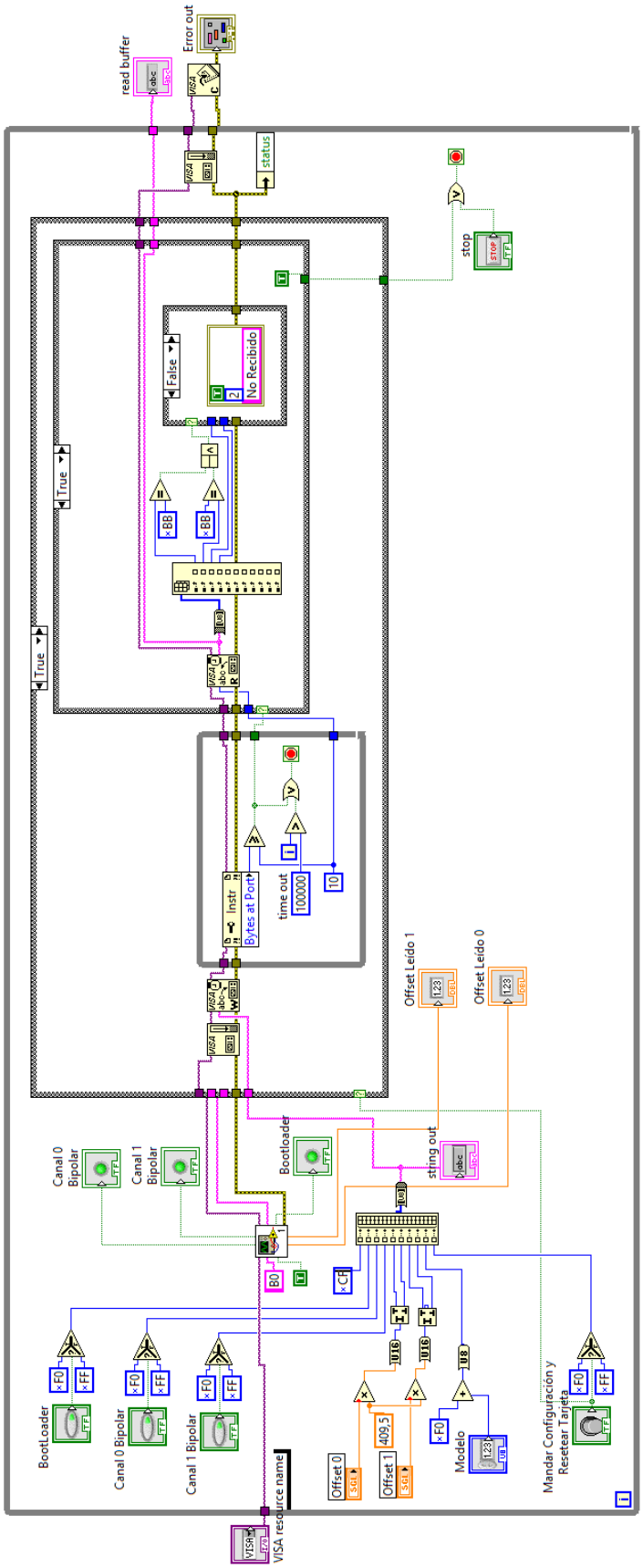


Figura 8. Programa .vi para almacenar los datos de *setup* en la TAD.

El comando que se elige para cambiar los datos almacenados será `CFh`, después se enviarán la siguiente información en este orden, si se ejecuta el bootloader en el caso de los PIC, si los canales de salida serán bipolares o unipolares, el offset de cada canal, que se habrá calculado previamente con algún programa de los anteriores y un polímetro u osciloscopio y la orden de si se resetea la tarjeta o no, válida para cargar el bootloader en el siguiente arranque en el caso de los PIC. Nótese que se hace uso del .vi anterior para pedir la configuración almacenada.

La parte siguiente son los indicadores de como se encuentra la tarjeta y el tratamiento de errores. Conviene explicar que se han elegido como "indicador de trama", una cadena de caracteres que sea imposible que se dé de otra manera que no sea intencionada. En el caso de la lectura analógica, se usa la cadena (en hexadecimal) `AXAA0YYYh`. Siendo la `X` el canal que se debe leer e `YYY` el dato leído de ese canal (la tensión). Puesto que la cadena `AA0Xh` es irrepetible, se toma por bueno cuando se recibe esta respuesta y en otro caso se desecha. En la petición de la configuración salvada se hace uso de la cadena `BBBBFX...h` en este caso la cadena `BBBBFh` es irrepetible también.

4.4 Programa test de las TAD

A modo de test, y de muestra a los futuros usuarios, se diseñó un programa .vi que usaba los canales tanto de entrada como de salida.

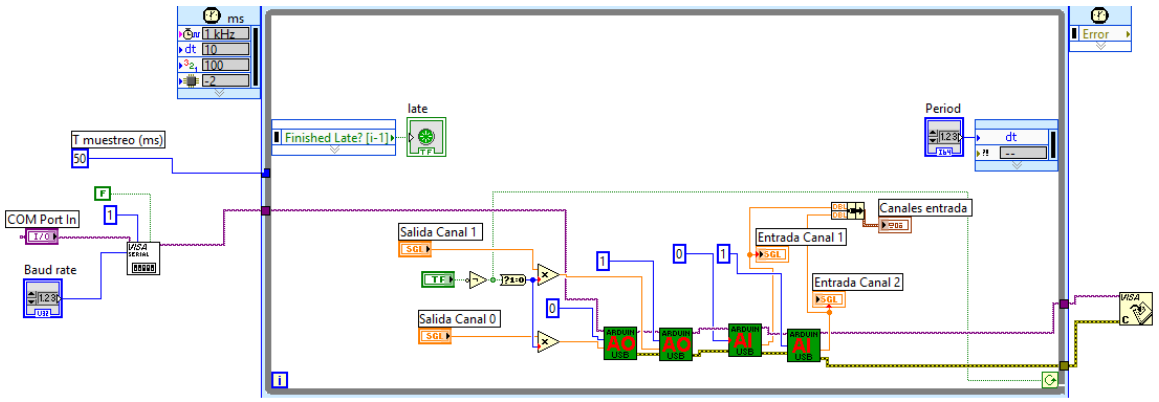


Figura 9. Programa de test para las TAD diseñadas.

Observamos en la Figura **9** como se usan los .vi anteriores a los que hay que pasarles el puerto serie donde lo podemos encontrar, el canal donde se debe leer o escribir el valor y, según se trate de la lectura o la escritura, el valor leído o a escribir. Los valores leídos se muestran en una pantalla tipo osciloscopio. La configuración del puerto se realiza antes de entrar al bucle con el VISA correspondiente. En esta VISA se configura que el bit a false que no hay carácter final en el envío de cadenas, pues por defecto sí lo hace. En la programación tanto de Arduino como de los PIC, se contempló la posibilidad de que este carácter existiera y se prescinde de él, pues no es necesario y puede provocar más problemas que soluciones.

4.5 Configuración del PID cuando se usa como controlador

Se realizó un programa .vi que sirve para cambiar los valores del PID integrado en el controlador de manera sencilla desde el PC. Dada la similitud con los anteriores, en esta ocasión se muestra solo la parte frontal.

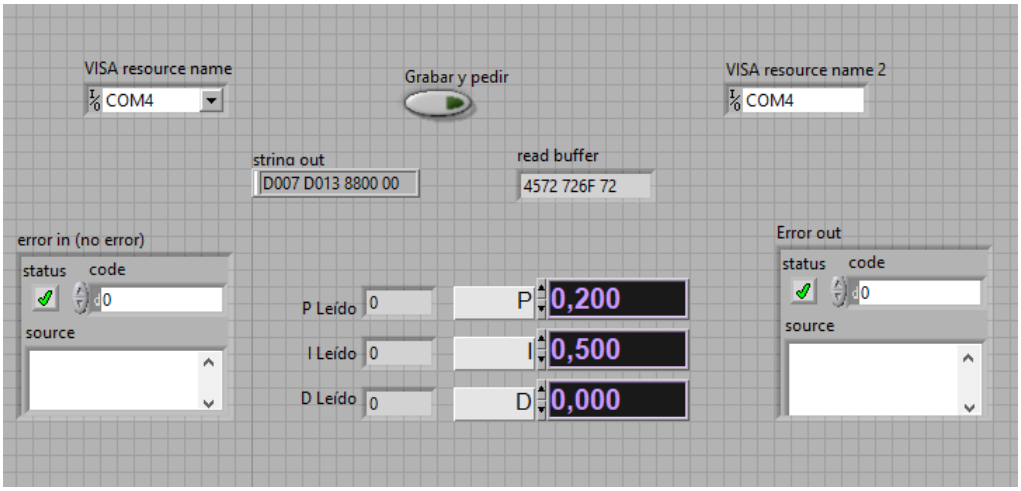


Figura 10. Programa usado para cambiar los valores del controlador PID interno.

No se diseñó ningún programa para la programación de los modelos, pues para ese caso es más sencillo programar el código directamente.

5 Comparación con tarjetas comerciales

Para saber si las TAD diseñadas cumplen con las especificaciones se realizaron varias pruebas y los resultados se compararon con las tarjetas de bus PCI existentes, concretamente la PCI9112 de ADLink y la PCI-1711 de Advantech. Las dos tarjetas son capaces de muestrear señales por encima de las cien mil veces por segundo sobre un canal, en caso de querer hacerlo sobre más canales, estas muestras se dividen por el mismo número. A la hora de generar la señal, el tiempo de conversión es de 8 μs [7], algo imposible de alcanzar para un PC funcionando con un S.O. de usuario normal. Se comprueba que lo máximo que se consigue funcionando con C++ es de muestras y salidas de 1 ms, mientras que con Matlab se logran 2 ms y con Labview se quedan en 5 ms. La mayoría de las prácticas se realiza con muestreos de 10 ms y las que no lo hacen con ese tiempo, se hacen con 100 ms.

Estas tarjetas tienen también otras opciones como temporizadores y salidas digitales, pero no se usan en las prácticas, por lo que en las TAD diseñadas no son el objetivo principal.

Se contempla usar la TAD comercial USB-6001 de National Instruments[8]. Esta también será la base a comparar, pues se ha decidido abandonar cualquier bus interno. Los resultados son buenos, la limpieza de la señal generada es correcta y se puede muestrear una señal de 100 hz sin problemas. La elección de los 100 Hz no es arbitraria, se recuerda que la mayoría de prácticas operan muestreando a 10 ms, que son los 100 hz.

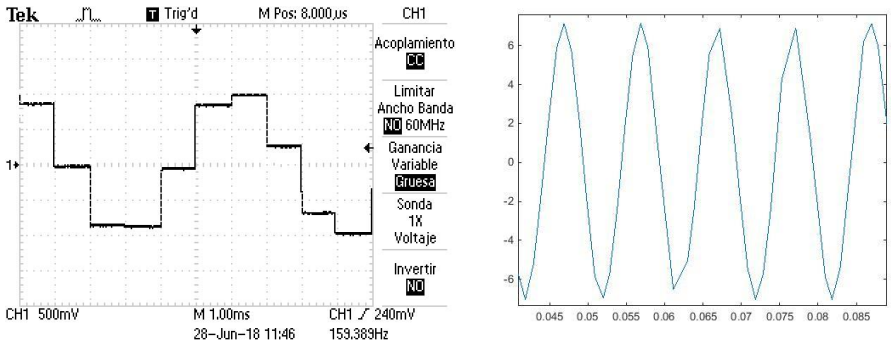


Figura 11. Senoidal generada (izquierda) y leída (derecha) con USB-6001 desde MatLab.

Hay que tener en cuenta también que estos datos son máximos y que el programa en el PC tendrá que leer la entrada, calcular la salida y enviar. Tampoco tiene problema al hacerlo. Se recuerda que esta tarjeta cuesta doscientos cincuenta euros con IVA y portes incluidos y ese será el precio que hay que reducir. Como dato, las de bus PCI o PCI *express* pueden superar los quinientos. Es cierto que tienen más opciones, pero luego no se usan, no son a valorar.

La TAD diseñada con un PIC a partir de una tarjeta ISA ACL8112 fue la que mejores resultados dio en cuanto a ruido de señal, impedancia de entrada y precisión obtenida. Esta tarjeta tiene especificaciones muy parecidas a las anteriores [1]. De hecho en la búsqueda de las PCI necesarias para sustituirla en su día se tuvo en cuenta que el hardware asociado disponible en el departamento fuera también compatible. En concreto hay una caja con conectores DB37 que incluye un circuito de protección para evitar daños en caso de contactos erróneos. Este DB37 marcó la elección de la PCI-9112. Se resalta este hecho pues se trata de economizar y proteger la tarjeta siempre es una buena idea y en las últimas TAD ya se contempla esa opción sin necesidad de incluir esa caja.

La velocidad de muestreo solo está limitada por el puerto USB y en menor medida por los programas que hacen uso de él. Los tiempos logrados son de 2 ms cuando la TAD se dedica por entero a ello y 5 ms cuando además el programa a ejecutar tiene que calcular y usar tanto la salida como la entrada.

La solución de TAD solo-PIC fue la que peores resultados obtuvo cuando se usó el PWM como generados de señal de salida, pues el rizado era de 50mV, muy elevado para ciertos procesos y para una cualquiera de las otras TAD construida con convertidores DA. En la Figura **12** se puede observar la señal obtenida. La tensión debería ser una recta de 0 V.

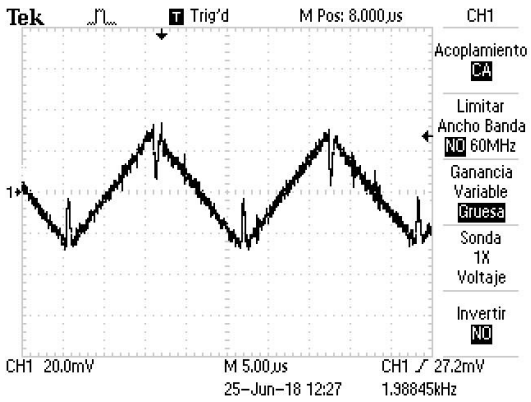


Figura 12. Salida PWM del PIC.

Respecto a la velocidad de muestreo, es inferior a la anterior, pero más que suficiente. No se encuentra otra limitación que la ya comentada del puerto USB. Se añade el DAC MCP4922 y la señal mejora de manera ostensible, siendo comparable a las anteriores. Esto no añade tiempo a las conversiones.

En la TAD que desarrolla con Arduino se encuentra el problema más grave hasta ahora, se trata de un nivel ruido muy alto en la señal de entrada. Se sospecha y confirma que viene derivado del convertidor DC/DC empleado para generar la tensión bipolar desde los 5 V que entrega el puerto USB. Se intenta reducir el problema con condensadores y resistencias de filtro, lográndolo a medias. La solución final es aplicar un filtro digital ejecutado por el procesador del propio Arduino. En la Figura **13** se muestran las señales antes y después de aplicar el filtro.

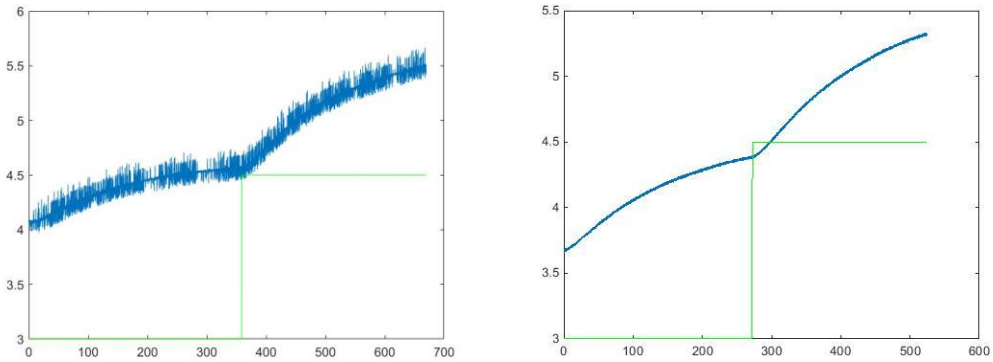


Figura 13. Lecturas de Arduino sin filtro y con filtro digital interno.

La discrepancia en los valores es debida a que la instantánea se tomó sobre el proceso que simula un horno y este se encontraba a diferentes temperaturas de partida.

Respecto a las velocidades de muestreo, se observa que es el que peor resultados obtiene. Se sospecha que debido al propio driver. En cualquier caso, es inferior a los 10 ms y sirve de sobra para todas las prácticas.

En el próximo capítulo se abordan diferentes prácticas resueltas con las TAD diseñadas

6 Pruebas de los desarrollos en las prácticas de laboratorio

En este capítulo se probarán los desarrollos anteriores en el funcionamiento real de varias prácticas, cada uno con un proceso y a su vez, cada uno con un programa de los usados de manera habitual.

Se sustituyen las líneas de código o los .vi que manejan las TAD internas por las líneas de código necesarias para que funcionen las nuevas con un éxito total. Los procesos se identifican y controlan del mismo modo que se hacía con las anteriores sin que se noten desplazamientos o fallos en la señal, ni sea necesario una aclaración al profesorado de lo que realiza cada módulo, pues su uso es bastante intuitivo.

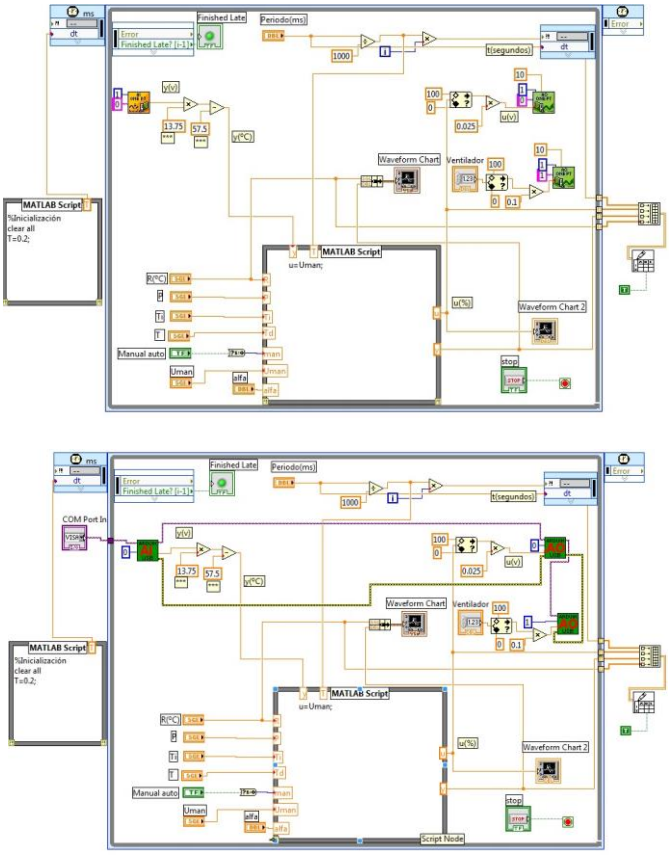


Figura 14. Sustitución en la práctica de control [13] de una célula Peltier de las .vi para las PCI9112 por las nuevas .vi

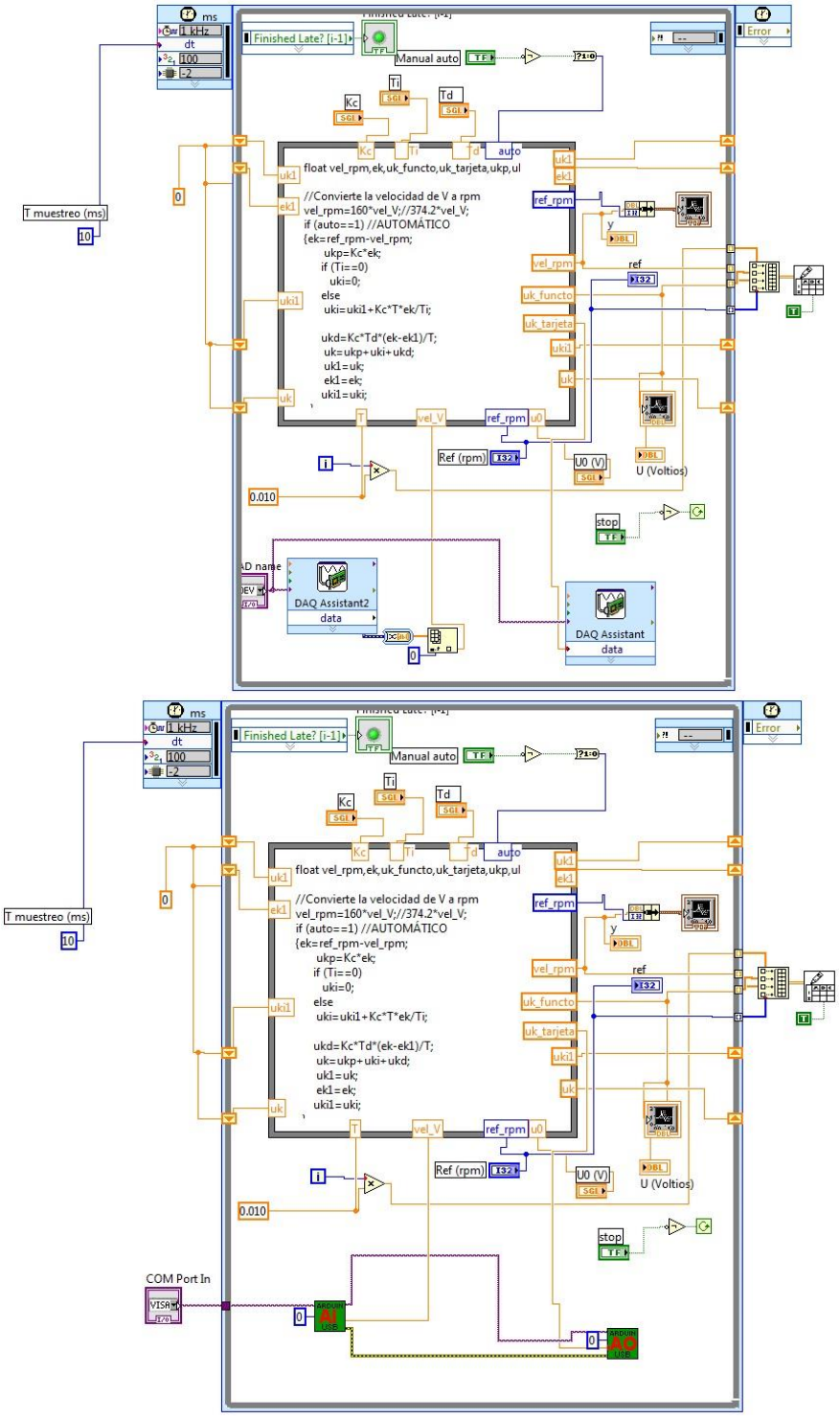


Figura 15. Sustitución en la práctica de control de un motor de CC [14] de las .vi necesarias para la tarjeta USB6001 por las .vi necesarias para la TAD con Arduino o PIC.

7 Simulador de procesos por Arduino

En este capítulo se implementará un modelo en el interior de Arduino para que comporte como un proceso cualquiera del que se conozca dicho modelo. Aunque es algo que se puede hacer en cualquiera de los anteriores, solo tiene sentido hacerlo en Arduino y en el solo-PIC, pues el otro sería muy aparatoso y no se podría alimentar desde un cargador USB o similar. Se descarta el uso del PIC pues Arduino posee más soporte.

Se hace notar que para usar de esta manera la TAD se necesita otra tarjeta o similar en el lado de control, es por eso que esta aplicación se aprovecha para usar sobre todo con los autómatas del departamento que tienen entradas y salidas analógicas, pues en algunos laboratorios no están presentes las fuentes de alimentación necesarias para actuar sobre los procesos reales. Como apunte, se recuerda que el diseño solo-PIC serviría para dotar a los autómatas sin entradas y salidas analógicas de ellas. No obstante ese uso sería minoritario.

Después de averiguar los modelos necesarios para la posición y velocidad de un motor Feedback 33-100 se realizan pruebas para con ellos y el resultado es satisfactorio, aunque se tienen que retocar los detalles para hacer compatibles las tensiones de ciertos autómatas. Por ejemplo el Modicon TS Micro de Schenider, que solo dispone de una salida de 0 a 10 V [16] y no permitiría el control de posición.

Posteriormente se implementa el proceso del horno y funciona perfectamente sin necesidad de retocar nada, al ser un proceso más lento. Para las pruebas de este proceso se usa el autómata de Modicon M241 de la misma marca que el anterior y dotado del módulo de entradas salidas analógicas [17]. El usar estos autómatas de entre todos los que tiene el departamento se debe precisamente a la ausencia de fuentes en el laboratorio donde están situados.

Como solución a la posibilidad de emular distintos procesos en la misma tarjeta, se decide introducir en Arduino varios modelos que la TAD ejecutará o no según la configuración programada previamente a través del programa de configuración. Para definir su uso, se usaría el programa de configuración para decidir el modelo a simular, se programaría y al reiniciar la TAD dejaría de comportarse como una TAD y pasaría a comportarse como ese modelo. Es similar al modo *bootloader* del PIC. Cuando se necesite, se alimenta con un cargador o con el propio PC y se conecta al autómata.

8 Integración de los controladores en la propia TAD

Al llevar integrado las placas un microcontrolador de suficiente potencia, es posible realizar el control de los procesos directamente sin necesidad de PC, solo sería necesario encontrar los valores PID del controlador y pasárselos a través de un programa. Después sería observar el comportamiento. Aunque este no es el objetivo inicial de la TAD, en algunas asignaturas de robótica se hace uso de este tipo de control embebido para actuar sobre motores de corriente continua.

Se observa que usar el controlador PID [15] en la propia tarjeta mejora notablemente los tiempos de adquisición y actuación, bajando del mili segundo. Para los procesos muestreados a 10 ms desde el PC es necesario cambiar el control y se trabaja lo necesario en el software de la tarjeta para que sea el propio procesador de la TAD el que muestree a alta velocidad y pase después los datos necesarios al PC, desde allí se realiza el modelo con MatLab. Posteriormente los resultados se pasan de nuevo a la TAD para cambiar el controlador.

Los resultados son buenos, pero no es el objetivo principal de la TAD y desde el departamento se decide que las asignaturas que lo necesiten incorporarán su propia placa de control.

9 Conclusiones

El objetivo inicial de estas tarjetas era sustituir de manera definitiva o por lo menos durante un tiempo razonable a las TAD comerciales, caras y dependientes de los fabricantes de dichas TAD en lo que respecta a los drivers y actualizaciones de ellos. Un problema común es pasar los sistemas operativos de 32 bits a 64 bits para aprovechar todas las características del hardware reciente. Esto provoca situaciones complejas de solucionar si la TAD tiene algunos años y el fabricante ha dejado de dar soporte para ella.

También se encuentra el problema de la compatibilidad de los buses internos, el bus ISA despareció hace tiempo de los PC de uso común y solo se encuentra disponible en PC industriales que, solo por serlo, son más caros, además de tener menos capacidad que cualquier PC de la época. Al bus PCI le pasa lo mismo, cuesta encontrar placas que lo lleven y las que lo llevan no son tan actuales como deberían, con lo cual se corre el riesgo de comprarlas y posteriormente no ser 100% compatibles con el S.O. a instalar.

Esto lleva a dos opciones, o bien usar una tarjeta comercial USB, como la mencionada NI-USB-6001 de National Instruments (NI) o bien fabricarse una tarjeta también USB. En las pruebas la tarjeta comercial responde bien y funciona con todos los programas usados, pero se tiene que hacer uso de los recursos software proporcionados por NI, quedando entonces a expensas de lo que este fabricante decida respecto a futuras compatibilidades.

El precio de esta tarjeta es inferior al coste de las internas, en parte porque tiene menos capacidades, menos entradas y salidas digitales, ausencia de temporizador… pero sigue siendo alto. Todas las tarjetas diseñadas en este TFG reducen el precio del material a casi la mitad en el peor de los casos. Pero hay que tener en cuenta que alguien tiene que montarlas y eso lleva tiempo o dinero de mano de obra.

Respecto a los drivers, superan a medias ese problema: se comportan como un puerto serie. Esto simplifica las cosas a la hora de instalarlas, solo será necesaria la instalación una vez y todos los programas con acceso a ese puerto, podrán manejarla. Pero en el caso de Arduino, se sigue dependiendo del driver que proporcione el fabricante. No pasa lo mismo con los PIC, pues el propio S.O. Lo reconoce como puerto serie sin drivers adicionales. Sí sería una pega si fuera necesario actualizar el *firmware* interno, lo que se conoce en el argot como *bootloader*, este problema se podría solucionar usando el conector ICSP (*In Circuit Serial Programming*, Programación serie en circuito). Este conector permite la programación del microcontrolador [3] sin necesidad de desmontarlo de placa. Pero para el caso que nos ocupa, sí sería necesario desmontar la caja en la que iría montado el circuito para tener acceso a dicho conector.

Las TAD diseñadas disponen además de diversas características no existentes en la TAD comercial: posibilidad de simular procesos, de diseñar controles PID embebidos, trabajo independiente del PC para usarse con los autómatas.

Sin embargo, hay que tener en cuenta el uso real de dichas características. Respecto a la simulación, puesto que la alimentación de la TAD seguramente vendría por el propio PC, podría ser este el que simulara el proceso y usara la TAD “normal” para lo mismo, se

podría acompañar también con algún programa que mostrara imágenes del proceso real, dando mejor apariencia. Luego no es una característica determinante.

El control embebido tiene sentido en las asignaturas que usan Arduino como base, pero no tanto en las demás, pues siempre se tiene que hace uso de un PC antes de poder controlar algo. Esto lleva a que se pierda un poco el sentido de lo que se está haciendo en una práctica de control usual, donde los parámetros se fijan directamente en el PC y no hay necesidad de conocer Arduino, o PIC en el caso de que se decidiera.

En definitiva, las TAD diseñadas cumplen con los requisitos iniciales, incluso con alguno más, pero es competencia del departamento decidir si se fabrican las placas o se opta por la solución comercial y esta decisión no siempre depende de características o velocidades de muestreo.

10 Bibliografía

- [1] ADLINK TECHNOLOGY INC. User's ManualACL8112PG v3.70 . 2004. https://www.contradata.it/images/mvc/MANUALI/774/User-Manual-ACL-8112PG-Rev3.70.pdf (fecha consulta 2/5/2018)
- [2] Gustavo Mercado. Introducción al bus ISA. 1999. http://www1.frm.utn.edu.ar/tecnicad2/tec_dig2/doc/Introduccion%20al%20Bus%20ISA.pdf
- [3] Microchip. PIC18F2550 Datasheet. 2006. http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf (fecha consulta 2/5/2018
- [4] Microchip. MPLAB C for PIC18 v3.47 Release Notes. 2014. http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB-C18_v3.47-README.html (fecha consulta 2/5/2018)
- [5] Linear Technology. LTspice IV getting started guide. 2011. http://www.analog.com/media/en/simulation-models/spice-models/LTspiceGettingStartedGuide.pdf?modelType=spice-models (fecha consulta 2/6/2018)
- [6] Miró Orozco, Ignacio. Introducción al Orcad 15.7. Video tutorías. 2011. https://www.youtube.com/watch?v=IW2d6x-6KYs (fecha consulta 3/5/2018).
- [7] ADLINK TECHNOLOGY INC. PCI9112 Datasheet. 2014. https://www.adlinktech.com/Products/Data_Acquisition/Multi-Function_DAQ/PCI-9112?lang=en (fecha consulta 23/6/2108)
- [8] National Instrruments. NI6001 Specifications. 2014. http://www.ni.com/pdf/manuals/374369a.pdf (fecha consulta 23/6/2018)
- [9] Fernando Giménez. Primeros pasos con Matlab. 2013. https://www.youtube.com/watch?v=Nu8Adto0ym8. (fecha consulta 4/6/2018)
- [10] Alfredo Villanova Moltó. Creación de programas con Labview. 2017. https://www.youtube.com/watch?v=xssnEOWdVtQ. (fecha consulta 18/5/2018)
- [11] Matlab. Arduino con MATLAB y Simulink, Parte 3: Programando Arduino con Simulink Carlos Hernández Franco. 2015. https://www.youtube.com/watch?v=0cos5FHEwHM (fecha consulta 6/6/2018)
- [12] Sebastian Nilsson. Blog DueFlasStorage. 2013. http://sebastiannilsson.com/en/blogg/ersattare-for-eeprom-pa-arduino-due/ (fecha consulta 14/7/2018)
- [13] Juan Manuel Herrero, Control de temperatura con Peltier. Prácticas de Laboratorio Automatización y Control 2107. Departamento de Ingeniería de Sistemas y Automática. Documento interno.

- [14] Xavier Blasco, Control de Motor de C.C. Prácticas de Sistemas Automáticos 2107. Departamento de Ingeniería de Sistemas y Automática. Documento interno.
- [15] Enrique Bernabeu, Apuntes clases teóricas Control por Computador. Departamento de Ingeniería de Sistemas y Automática. 2017. Documento interno.
- [16] Schneider Electric. Connection interfaces TSX. 2001. http://download.schneider-electric.com/files?p_Doc_Ref=Micro_Telefast-EN&p_EnDocType=Catalog&p_File_Name=Micro_Telefast-EN.pdf (fecha consulta 19/5/2017).
- [17] Schneider Electric. Guia de hardware Modicon M241. 2018. https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=EIO0000001459.07.pdf&p_Doc_Ref=EIO0000001459 (fecha consulta 15/6/2018)