Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Diseño de una tarjeta de adquisición de datos basada en Arduino

Autor:

López Villanueva, Ricardo

Tutor:

Zamarreño Cosme, Jesús María
Ingeniería de Sistemas y Automática

Valladolid, septiembre de 2021.

RESUMEN

Con este proyecto se busca desarrollar un dispositivo de medición y escritura de valores que permita sustituir las tarjetas de adquisición de datos instaladas en el laboratorio de control de procesos. Este dispositivo se consigue a partir de un microcontrolador basado en la plataforma Arduino, con la ventaja de tener un precio menor y ofrecer unas prestaciones similares.

La comunicación entre la tarjeta y computador será a través de la tecnología OPC, la cual necesita de un servidor que lo gestione y 'Software Tools 4 Makers' ha desarrollado uno específico para esta aplicación. A través del código introducido en el microcontrolador Arduino, similar a C++, se establecen todos los parámetros necesarios para una comunicación satisfactoria.

Con todo ello, el prototipo final permite al usuario el control y supervisión de la instrumentación, con la posibilidad de recoger las señales de diferentes sensores y enviar la adecuada a los actuadores.

Palabras claves

Arduino

Tarjeta Adquisición de Datos

Comunicación OPC

C++

Instrumentación

ABSTRACT

The aim of this project is to develop a device for measuring and writing values to replace the data acquisition cards installed in the process control laboratory. This device is founded on a microcontroller based on the Arduino platform, with the advantage of having a lower price and offering similar features.

The communication between the board and the computer will be through OPC technology, which needs a server to manage it and 'Software Tools 4 Makers' has developed one specifically for this application. Through the code introduced in the Arduino microcontroller, similar to C++, all the necessary parameters for a successful communication are established.

With all this, the final prototype allows the user to control and supervise the instrumentation, with the possibility of collecting signals from different sensors and sending the appropriate one to the actuators.

KeyWords

Arduino

Data Acquisition Device

OPC Communication

C++

Instrumentation

AGRADECIMIENTOS

A mi familia, por no limitar mis decisiones y permitirme crecer.

A mi pareja y compañera de vida Marta, por apoyarme en todo momento, hacerme feliz y no permitir que lo dejase de intentar. Sin ti no habría sido posible.

A mis profesores y especialmente mi tutor Jesús María Zamarreño Cosme, por exigir siempre un poco más de mí, potenciarme y guiarme en el camino.

Gracias a mis compañeros, amigos y a toda persona que me he encontrado a lo largo de este camino, todas habéis aportado vuestro granito de arena.

Y por último gracias a mí. Gracias por creer en ti, gracias por trabajar duro y gracias por no rendirte.

ÍNDICE GENERAL

ÍNDICE ILUSTRACIONES

ÍNDICE TABLAS

1. INTRODUCCIÓN

En la industria, como en muchos otros sectores, el control de los procesos que se llevan a cabo dentro de una empresa es y ha sido uno de los puntos clave dentro de la viabilidad y desarrollo de la misma. Más en concreto, en el sector industrial, el control del proceso productivo ha sido la clave para mejorar la fiabilidad, reducir los costes y aumentar la producción, por lo que siempre ha sido medido y supervisado de una forma u otra. Cuando no existían medios, se realizaba de forma manual, con operarios al cargo pendientes de los cambios o errores más relevantes para realizar la acción correctiva correspondiente, con todo lo que esto supone. Con el tiempo y la llegada de computadores lo suficientemente potentes, esta labor de control podía llegar a delegarse, por lo que fueron automatizando cada vez más.

El avance global y la llegada de circuitos integrados aumentaron las prestaciones de forma exponencial, hasta llegar a los dispositivos inteligentes y controladores conectados a red, capaces de comunicarse entre sí, muy parecidos a los de la actualidad que pueden recoger toda esta información de una forma más precisa, rápida y continuada en el tiempo, transmitiéndola a un servidor capaz de almacenarla y procesarla.

Actualmente, toda empresa dedicada al sector industrial tiene todo el proceso de producción controlado, parametrizado y monitorizado para conseguir los mejores resultados posibles y evitar errores catastróficos que supongan la paralización de la fabricación, todo gracias a los avances tecnológicos actuales.

El objetivo del control de procesos industriales es obtener un producto final con unas características determinadas que cumpla con las especificaciones y niveles de calidad exigidos. Estos sistemas se componen a nivel de campo de sensores, medidores de las variables a controlar; controladores, responsables de calcular la acción de control, y elementos de control como válvulas, que reciban la señal del controlador y actúen directamente. Estos elementos necesitan transmitir la información a un computador que sepa cuál es la acción de control correspondiente en cada caso; por lo tanto, es necesario de un elemento que acondicione estas señales y las trate desde un entorno analógico a otro digital sin perder información, así como los correspondientes protocolos de comunicación que la regulen en todo momento.

La transmisión de información puede ser muy variada, con distintos protocolos que se pueden utilizar y distintos métodos para recoger las señales en cuestión. Como tal no es sencilla; es necesario un estándar entre emisor y receptor, un mismo lenguaje que posibilite acceder a los datos de cualquier fuente. En el caso de estudio actual y un método muy usual en el sector industrial es el estándar OPC, el cual ofrece una interfaz común entre estos componentes permitiendo su interacción, con lo que un servidor OPC será la fuente de los datos y cualquier aplicación basada en el mismo protocolo puede acceder a él. En resumen, permite la comunicación entre el origen y el destino de la información si ambos están basados en este mismo estándar. Es además el protocolo de comunicación disponible en las plantas del laboratorio de control automático del departamento de

Ingeniería de Sistemas y Automática, en la residencia Alfonso VIII, sede de toda prueba realizada para la consecución de este proyecto y objetivo final del mismo.

La última cuestión derivada de la comunicación o transmisión de información es el dispositivo de recolección de señales. Una muy común es la instalación de tarjetas de adquisición de datos (TAD), que como su propio nombre indica, mide las señales en cuestión y las convierte a digital con la menor pérdida de información posible para su lectura e interpretación en el computador. Están compuestas de diversos componentes software programables en PC y hardware de medición, con su correspondiente acondicionamiento.

El objetivo de este proyecto recae en este mismo punto, conseguir sustituir este sistema con unos costes elevados para un fin académico por un microcontrolador globalmente conocido con el soporte y posibilidades que esto supone. Para ello es necesario, además, el desarrollo de un servidor OPC particular, ya que el objetivo es sustituir un componente por otro sin necesidad de cambiar el protocolo y así beneficiarse de todo el software industrial orientado a este estándar de comunicación. Además, se va a proceder a realizar con la placa más sencilla y más conocida de este fabricante, que es el Arduino UNO, a la cual habrá que aplicar una serie de configuraciones e instalar una serie de componentes adicionales para llegar a cumplir las especificaciones y requerimientos de calidad esperados para un dispositivo de este calibre.

La idea proviene de que ambos poseen dispositivos de medición de señales analógicas, a distinto nivel y resolución pero con un mismo fin, por lo que se puede conseguir la misma labor con un microcontrolador de plataforma libre como Arduino que con un sistema de adquisición de datos, el cual está orientado a ambientes más complejos y ruidosos los cuales no son habituales en un laboratorio con un fin académico, con todos los sobrecostes que esto supone.

1.1. OBJETIVOS

El objetivo de este proyecto es conseguir con un microcontrolador como Arduino UNO, el más común y usual con todas las ventajas que esto conlleva, conseguir sustituir a una tarjeta de adquisición de datos cualquiera, en este caso las instaladas en el laboratorio de laboratorio de control automático del departamento de Ingeniería de Sistemas y Automática, la TAD DAQ-USB-1408FS-Plus de la empresa Measurement Computing. Para ello, el proyecto se apoyará sobre un servidor OPC ya desarrollado, beneficiándose de todas las ventajas que esto supone en cuanto a soporte y plataforma. Este software libre se denomina “OPC SERVER FOR ARDUINO/GENUINO”, desarrollado por Ildefonso Martínez Marchena [1], el cual ofrece soporte y ayuda a todo el que utilice su configuración. Por lo tanto, una vez configurado el dispositivo mencionado para la lectura y escritura con el software y hardware adecuado, se puede llegar a conseguir unas prestaciones similares al dispositivo comercial mencionado.

En el caso de estudio objetivo, las plantas del laboratorio mencionado presentan tarjetas de adquisición de datos con multitud de entradas, con una precisión elevada, por lo que están totalmente desaprovechadas para un uso educativo con el consiguiente gasto económico que esto supone. En este laboratorio se cuenta con varias plantas de medición de nivel de depósito, las cuales suelen constar de como máximo 4 transmisores (nivel, caudal…) y 2 actuadores (bombas…), lo que son 4 entradas y 2 salidas hacia nuestro sistema de adquisición de datos. El sistema actual presenta 8 entradas y 2 salidas analógicas diferenciales, junto con 16 E/S digitales, con una resolución de 14 bits. Podemos recurrir a este microcontrolador con una precisión muy parecida y adaptado al entorno de trabajo por un precio mucho más reducido.

1.2. ESTRUCTURA MEMORIA

La memoria actual del proyecto desarrollado se puede dividir en 8 partes o capítulos, además de la presente introducción, donde se expone pormenorizadamente los pasos seguidos, así como las características de todos los componentes utilizados:

- ***Capítulo 2*** - *Arduino*: descripción al detalle del microcontrolador implantado, con sus características técnicas y sus requerimientos para un correcto funcionamiento, así como los componentes a utilizar para conseguir las especificaciones correctas.

- ***Capítulo 3*** - *Tarjeta de adquisición de datos*: descripción de características y componentes del dispositivo a sustituir, con pruebas y detalles técnicos tanto de software como de hardware.

- ***Capítulo 4*** - *Funcionamiento estándar OPC*: repaso y resumen del estándar en cuestión utilizado, así como su implantación en el microcontrolador Arduino con el software desarrollado por st4makers.

- ***Capítulo 5*** - *Desarrollo Hardware y Software*: detalle de los componentes necesarios instalados en el proyecto y su programación, alternativas y soluciones adoptadas, así como componentes estéticos diseñados y fabricados con un presupuesto final.

- ***Capítulo 6*** - *Instalación*: detalle de la instalación equipada en los equipos del laboratorio compuesta por el prototipo Arduino y sus componentes, el Arduino OPC Server y el software en el ordenador destino.

- ***Capítulo 7*** - *Validación*: pruebas realizadas para estudiar la viabilidad y validez del prototipo creado, así como su comparación con la TAD actualmente instalada.

- ***Capítulo 8*** - *Conclusiones y mejoras*: líneas futuras no implantadas por donde se podría desarrollar el proyecto a futuro que se considera mejorarían el prototipo realizado, así como conclusiones finales sobre él.

Para terminar, se incluye un apartado de referencias, webgrafía y bibliografía donde se incluye toda la documentación oficial de archivos y páginas web que hayan sido de ayuda para aportar la información necesaria y complementaria para el completo desarrollo del proyecto.

2. TARJETA ADQUISICIÓN DE DATOS

En el primer capítulo de introducción se ha mencionado que en este proceso de control de procesos es necesario un dispositivo que sea capaz de captar las señales, acondicionarlas al medio y convertirlas a código digital para que un PC sea capaz de interpretarlas. A este tipo de dispositivos se les denominan DAQ (Data AcQuisition) y existe una gran variedad entre los que escoger. Esta función en el laboratorio de control automático del departamento de Ingeniería de Sistemas y Automática la cumple la tarjeta de adquisición de datos MCC USB-1408FS-Plus [2], con la que se recogen los datos de varias plantas de control de nivel.

La tarjeta de adquisición de datos es un dispositivo hardware que actúa como interfaz entre un equipo informático y las señales físicas de sensores en planta. Esta transforma los códigos analógicos que ofrecen los instrumentos en señales digitales interpretables por una computadora a través de un convertidor analógico-digital. Además de este componente, suelen estar provistas de un circuito acondicionador de señales para eliminar ruidos y un bus conector con el ordenador. Todas estas características son redundantes en cualquier otro tipo de dispositivo DAQ.

El circuito acondicionador es preciso para eliminar la inestabilidad característica de este tipo de señales y ser posible su lectura por parte del convertidor. Para ello, este circuito tiene la capacidad de amplificar o reducir la magnitud de la señal en base al valor correcto, filtrarla eliminando sobrepicos impropios de la señal manteniéndolos estables y aislando perturbaciones del exterior que afectan a la recopilación de las mismas.

El convertidor por su parte es, como ya se ha comentado, el traductor o transformador de la señal original; permite su interpretación por parte del computador el cual se encarga de explotar los datos recogidos. A él llega la señal analógica filtrada por el circuito acondicionador, la transforma a una señal digital y la envía al dispositivo receptor para su lectura. Esto es por la concatenación de 3 procesos (Figura 1): el muestreo, la cuantificación y la codificación. Con el primero se toman valores cada cierto tiempo de la señal analógica, con el segundo se le asigna un valor numérico decimal y con el tercero se codifica en el sistema binario. Todo esto ofrece al computador una serie de 0s y 1s para su interpretación.

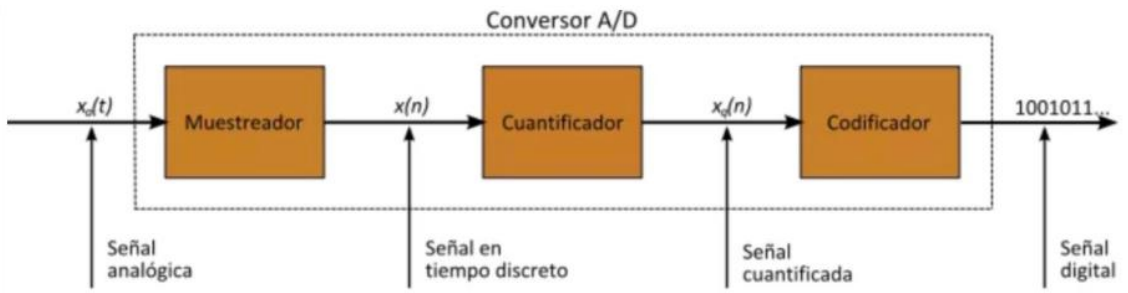


FIGURA 1: ESQUEMA CONVERSOR ANALÓGICO DIGITAL – HTTPS://WWW.WIKIWAND.COM/ES/CONVERSOR_DE_SE%C3%B1AL_ANAL%C3%B3GICA_A_DIGITAL

La tarea del bus hacia el computador, por lo tanto, es detectar esta señal digital binaria para interpretarla nuevamente y llevarla sin pérdidas al destino final que es el PC.

2.1. ESPECIFICACIONES TAD MCC USB-1408FS-Plus

El apartado en cuestión se centrará en concretar y nombrar cada una de las características técnicas más relevantes del sistema de adquisición de datos instalado en el laboratorio de automática, con el fin de comparar a posteriori con el dispositivo elegido para el proyecto.

En este caso, como ya se ha comentado, se trata de una tarjeta de adquisición de datos diseñada y fabricada por la compañía norteamericana 'Measurement Computing' [3]. Esta empresa provee una gran variedad de soluciones orientadas a la adquisición de datos, tanto de hardware como de software, junto a un soporte técnico y controladores para asegurar la funcionalidad de los mismos.

Dentro de la adquisición de datos, hay una gran variedad de sistemas y opciones a elegir:

- Registradores de datos: dispositivo captador de datos con el objetivo de formar un registro a lo largo de un periodo de tiempo. Estos pueden ser desde mediciones de temperatura hasta humedad, cualquier tipo de señal de interés a almacenar. Posee un procesador integrado, almacenamiento local y un software predefinido para ejecutarse como dispositivos independientes, haciéndolos portables y fáciles de usar.

- Dispositivos de adquisición de datos: categoría a la que pertenece el dispositivo en cuestión. Contiene un acondicionamiento y un convertidor A/D en su interior para llevar a cabo su función con la desventaja de la necesidad imprescindible de estar conectado a un PC. Estos son muy flexibles ya que pueden utilizar software ya predefinidos o por el contrario decantarse por un entorno de programación como Python, además una multitud de opciones para el bus de conexión lo que hace al DAQ una solución altamente personalizable.

- Sistemas modulares de adquisición de datos: diseñados para sistemas complejos con un número alto de canales que integrar y sincronizar con cada sensor origen. Al ser tan complejos de integrar los hacen muy flexibles, están hechos a medida, lo que incrementa su coste a costa de aportar soluciones que solo este sistema puede proporcionar.

Este modelo en concreto se encuentra dentro de la gama low cost de la compañía y en la actualidad aparece definido como un dispositivo DAQ con las siguientes características principales:

- 8 entradas analógicas, 4 diferenciales
- Resolución de 14 bits
- Frecuencia de muestreo máxima de 48KS/s
- 2 salidas analógicas
- 16 E/S digitales
- Salida digital de 6mA/24mA
- 1 entrada contador que permite contar eventos
- Soporte DAQami, Android, Linux y Matlab

Todo esto por unos 275$ (en España se puede conseguir por unos 370€), por lo que se encuentra dentro de la gama media de dispositivos DAQ multifunción de Measurement Computing. Estos componentes están orientados a aplicaciones de recuento de canales bajos a medios, contando toda la gama de entradas analógicas junto con E/S digitales y funciones de contador. Este dispositivo se alimenta con la conexión USB estándar de 5V con el PC, no requiere de ninguna otra alimentación, siendo compatible con USB 1.1 y USB 2.0. En el primer caso la velocidad puede verse limitada por el protocolo utilizado en cuestión.

FIGURA 2: ASPECTO DISPOSITIVO VISTA PLANTA SUPERIOR

En la figura 2 se puede apreciar la conexión USB con la computadora, los terminales de entrada y salida en su borde y el led de estado en su centro. Este último tiene la función de indicar cuando el dispositivo está conectado al PC con una luz verde fija o cuando se están transfiriendo datos con un parpadeo continuo. Los terminales proporcionan las siguientes conexiones:

- CH 0 IN -- CH 7 IN : 8 conexiones de entrada analógica
- D/A 0 -- D/A 1: conexiones salida analógica.
- Puerto A0 -- A7 y Puerto B0 -- B7 : 16 conexiones E/S digitales
- TRIG_IN: disparo externo
- CTR: contador externo
- SINCRONIZE: terminal bidireccional para reloj externo o sincronización de múltiples unidades
- +V0: Potencia de salida
- AGND Y GND: tierra analógica y tierra

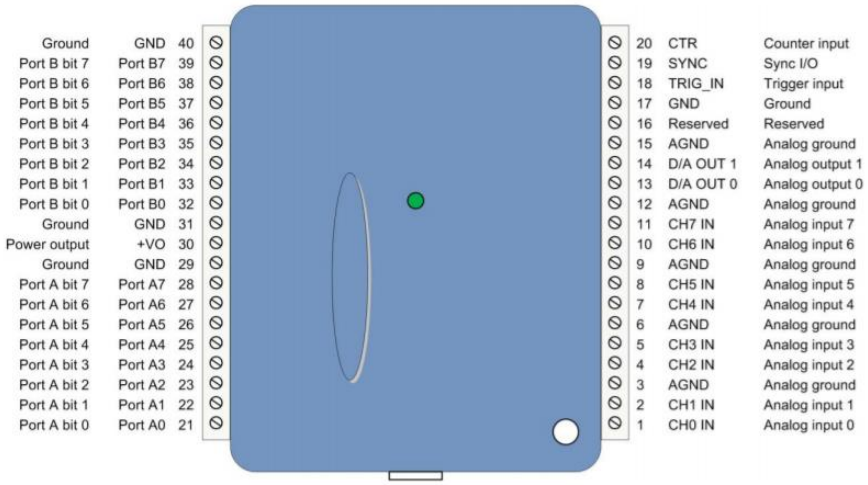


FIGURA 3: DISTRIBUCIÓN PINES TAD

Las entradas analógicas también pueden funcionar en modo diferencial, con la única diferencia que las entradas analógicas independientes serán parejas de conexiones diferenciales (CH0 IN / CH1 IN ⟶ CH0 IN + / CH0 IN LO -).

Por lo tanto, se puede concluir que este dispositivo se puede configurar los canales de entrada analógica como ocho canales de un solo extremo o cuatro canales diferenciales, en cuyo caso pasará a tener 14 bits de resolución. Cuando se configura para el modo de un solo extremo la resolución pasa a 13 bits debido a las restricciones que impone el convertidor A/D. En el modo diferencial se debe cumplir que cualquier entrada analógica permanezca en el rango de -10V a +20V con respecto a tierra, al igual que el voltaje diferencial máximo.

Las salidas analógicas presentes en la tarjeta en cuestión son dos conexiones en los puertos 13 y 14, los cuales se pueden controlar a velocidades de hasta 50000 actualizaciones de valor por segundo. Estos tienen una resolución de 12 bits y el rango de salida en este caso es de 0V a 5V.

Además, el dispositivo tiene 16 canales DIO configurados como dos puertos de 8 bits, el puerto A y el puerto B en la parte izquierda de la figura, desde el 21 al 40. El puerto B en este caso es de alta velocidad, pudiéndose conectar hasta ocho líneas DIO en cada uno de estos puestos. Estos terminales están configurados para leer falso cuando existe un voltaje nulo y verdadero cuando el voltaje es máximo, 5V. Todas estas líneas se elevan a USB +5V con una resistencia determinada, cambiando la configuración entre pull down o pull up con puentes internos.

Otra reseña a comentar es que la conexión del dispositivo a una computadora consume <100mA y cuando se ejecutan aplicaciones se puede llegar a extraer <500mA. La corriente de salida máxima disponible en el terminal de salida de potencia es de 100mA.

3. ARDUINO

3.1. INTRODUCCIÓN

Arduino [4] como concepto no es el dispositivo sino una plataforma de electrónica, la cual se basa en la filosofía de ofrecer un software libre a partir de unos componentes hardware y software sencillos de usar. Esta herramienta está compuesta por una placa de circuito impreso cuyo núcleo es el microcontrolador y un entorno de desarrollo diseñado para facilitar el uso de la electrónica en proyectos multidisciplinares.

Los microcontroladores forman parte de nuestro día a día, están presentes en la mayoría de los aparatos domésticos nutriéndose de sensores para captar señales del mundo físico y de actuadores para interactuar con él, es decir, leen de los sensores y escriben sobre los actuadores. El hecho de ser una plataforma abierta facilita su programación, abriéndose a la aportación de toda persona interesada en su desarrollo aumentando así sus posibilidades de forma exponencial.

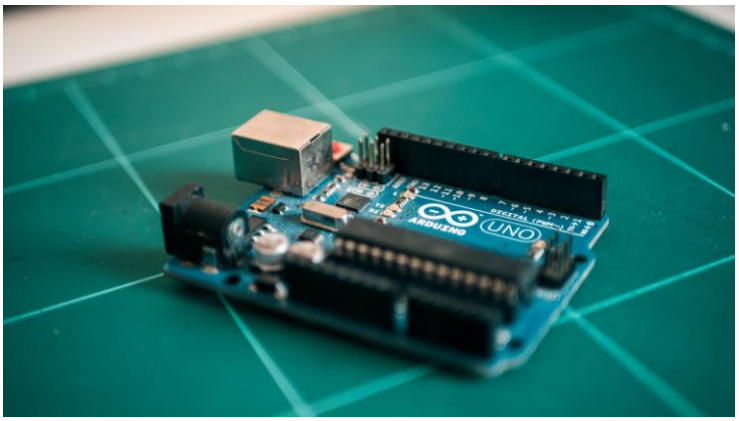


FIGURA 4: HARDWARE ARDUINO

El origen de la misma se remonta a comienzos de 2005, donde un grupo de estudiantes del Instituto IVREA en Italia que usaban microcontroladores con costes elevados se cuestionaron la existencia de una forma más barata con los que desarrollar sus proyectos en las aulas. La idea original fue fabricar una placa para uso interno de la escuela, pero cuando esta cerró se vieron obligados a liberar y abrir el proyecto a todo el mundo para que pudiesen participar en el desarrollo del mismo

El hardware de Arduino se compone de una PCB sobre la que se implementa un determinado diseño para que el usuario final no deba preocuparse de las conexiones eléctricas que necesitaría el microcontrolador para funcionar. El núcleo es este microcontrolador, el cual generalmente en las versiones oficiales es de la marca Atmel AVR, al cual se le añaden numerosos puertos de comunicación, entradas y salidas para conectar con dispositivos externos a él. Los modelos más usados son los Atmega168, Atmega1280, Atmega328 y Atmega8, caracterizados

por su sencillez, aunque la demanda actual se dirige hacia microcontroladores con arquitectura ARM de 32 bits.

Por otra parte, Arduino nos proporciona un software basado en un entorno de desarrollo IDE, que implementa el lenguaje de programación de Arduino y las herramientas apropiadas para transferir el firmware al microcontrolador junto con el bootloader ejecutado en la placa. Este código presenta sus particularidades pero está basado en el lenguaje de programación C++, el cual destaca por su sencillez y facilidad de uso haciendo este proceso de adaptación más liviano para cualquier usuario novel que quiera crear sus propios proyectos con Arduino.

Arduino por lo tanto promete ser una forma sencilla de realizar proyectos interactivos para cualquier persona, pero esta facilidad de uso y comunidad lo ha catapultado a su utilización en numerosos ámbitos donde no se pensaba que podría llegar. Esto se debe principalmente a su software libre y extensible sobre el que cualquiera puede ampliar y mejorar el diseño o el entorno, provocando un ecosistema de placas no oficiales para todo tipo de propósitos y de librerías de terceros que se adaptan a todo tipo de necesidades. Basta con adquirir uno de estos modelos tanto oficiales como extraoficiales para encontrar multitud de proyectos que poder implantar en la tarjeta y aprovechar, o por el contrario crear tu propio código para desarrollar una idea propia que se tenga en mente. Los principales motivos que hacen a un usuario en concreto decantarse por adquirir una placa con software Arduino son:

- Su comunidad: el alcance que le da su gran comunidad a nivel global lo dota de gran cantidad de documentación de gran relevancia y extensa que abarca cualquier necesidad.

- Entorno de programación multiplataforma: el cual se adapta a todo tipo de sistemas operativos.

- Lenguaje de programación fácil e intuitivo: basado en C++, lo hace fácil de comprender y captar, permitiendo la entrada a nuevos programadores y a la vez con gran capacidad para que los programadores más avanzados desarrollen todo su potencial.

- Bajo coste: la placa por excelencia estándar ronda los 20$, incluso uno mismo la podría fabricar, una ventaja de ser software libre.

- Versatilidad y reciclaje: una vez desarrollado un proyecto se puede volver a utilizar en el siguiente, es muy fácil desmontar los componentes externos y comenzar el nuevo proyecto. Todos los pines son accesibles al exterior a través de conectores hembra para sacar partido a todas las posibilidades.

La primera gran decisión para comenzar un proyecto con este software es la adquisición de una placa. Existen diversos modelos oficiales adaptados a todo tipo de necesidades, y como se ha comentado, también existe la posibilidad de crearla propiamente con los componentes que se adapten más al fin de la misma. Se pueden destacar cuatro de las más usuales en el mercado desarrolladas por el fabricante oficial:

- Arduino UNO: es el modelo básico para, a priori, cualquier tipo de proyecto. Como ya se ha comentado ronda los 20$ en la página oficial y tiene suficientes entradas y salidas para los primeros proyectos. En el caso del que se describe en esta memoria, ha sido la elegida por su variedad y características concretas que se describen en el siguiente punto de la memoria.

- Arduino Zero: si el usuario tiene necesidades que requieren más potencia esta placa es la ideal. El precio aumenta hasta los casi 35$ pero supera con creces a la anterior en CPU, RAM y memoria interna.

- Arduino Mega: si por el contrario lo que se busca son número de entradas digitales este es el modelo adecuado. Con un precio parecido al anterior, ronda los 35$, ofrece hasta 54 entradas digitales para conectarse a multitud de dispositivos externos. Esto a la larga va a dotar al usuario de más posibilidades con las que jugar en sus proyectos.

- Arduino DUE: aumenta las prestaciones respecto a la UNO con la principal característica de poseer 2 DAC para la conversión digital analógica. Esta ha sido una de las características que la han hecho a priori una candidata para este proyecto, pero que se ha descartado por motivos que posteriormente se comentarán.

Existen hasta 12 modelos oficiales y multitud de placas compatibles con este software, pero los más destacados se han expuesto con las características más importantes que los definen. Como también se comenta, hay gran variedad de marcas que han desarrollado su propia placa con características más o menos parecidas a las originales con precios más reducidos y con especificaciones muy interesantes. En el caso del proyecto que se describe en esta memoria, el utilizado ha sido el más básico de la gama oficial, el Arduino UNO, por una serie de características propias y especificaciones que hacen de esta placa la más adecuada para el proyecto a desarrollar. Estos motivos se describen en el siguiente apartado junto con las diferentes alternativas.

A continuación, se muestra una tabla con las características más relevantes de estos modelos para una comparación más intuitiva donde apreciar mejor las diferencias, ventajas e inconvenientes de los cuatro modelos comentados hasta el momento:

	ARDUINO UNO	ARDUINO ZERO	ARDUINO MEGA	ARDUINO DUE
MICROCONTROLADOR	ATmega328P	ATSAMD21G18, 32-Bit	ATmega2560	AT91SAM3X8E
VOLTAJE FUNCIONAMIENTO	5V	3.3V	5V	3.3V
PINS I/O DIGITALES(PWM)	14(6)	20(10)	54(15)	54 (12)
PINES ENTRADA ANALÓGICA	6, 10 bits	6, 12-bit	16, 10 bits	12, 12 bits
PINES SALIDA ANALÓGICA	0	1, 10-bit	0	2, 12 bits
MEMORIA FLASH	32KB	256 KB	256 KB	512KB
VELOCIDAD RELOJ	16MHz	48MHz	16MHz	84MHz
PRECIO	20 €	32 €	35 €	35 €

Tabla 1: Comparación modelos Arduino

3.2. ESPECIFICACIONES Y VENTAJAS ARDUINO UNO

Como ya se ha desvelado, la tarjeta o placa elegida para la consecución del objetivo es Arduino UNO, la más básica pero a la vez la más económica y que por otra parte mejor cumple con las necesidades. Existen varios requisitos para la conexión en concreto con la planta del laboratorio de Sistemas y control:

- Posibilidad de aportar señales analógicas, es decir, que posea un conversor digital/analógico para poder actuar sobre la bomba de agua que posee la planta.

- Medición de señales entre 0-5V; los sensores son industriales y son el voltaje con el que se comunican.

La primera especificación nos llevaba directamente a la placa Arduino DUE, es una de las que tiene DAC incorporado y además dos puertos frente al único que tiene la placa Arduino Zero. El obstáculo en esta decisión es el voltaje de funcionamiento que posee, el cual afecta directamente al rango de estas salidas analógicas que se verían limitadas a 3.3V. La solución pasa por crear un circuito acondicionador que aumentase esta tensión de funcionamiento, un circuito con muchos componentes y complejo que aumenta notablemente el precio de la misma o por el contrario crear la placa a partir de Arduino UNO, la cual tiene un voltaje de funcionamiento óptimo de 5V por lo que no es necesario este circuito salvando el inconveniente del DAC añadiéndolo como un dispositivo externo. Con este cambio reducimos el precio del prototipo final y su complejidad.

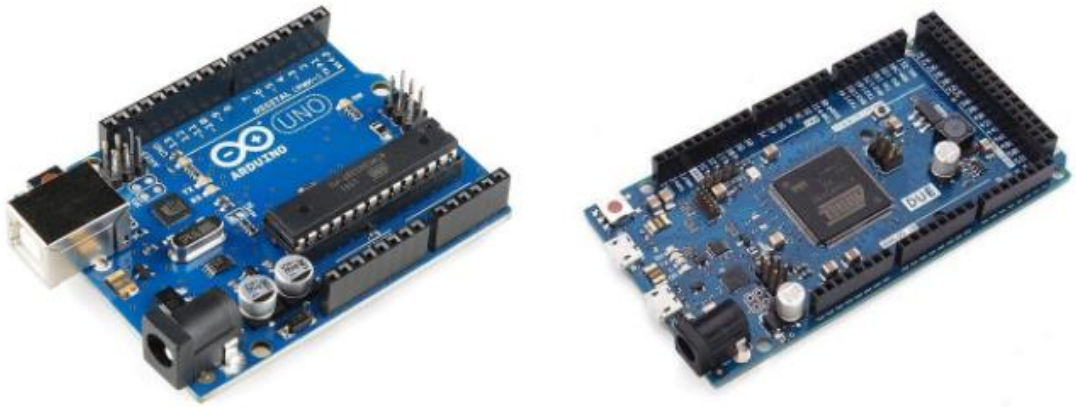


FIGURA 5: ARDUINO UNO A LA IZQUIERDA FRENTE A ARDUINO DUE A LA DERECHA

Otra opción valorada con unas características muy similares y en muchos aspectos más favorables son las tarjetas del fabricante Esspresif [5], más en concreto el modelo esp32. Estas entran dentro de la gama de chips compatibles con Arduino pero no oficiales que se hicieron populares por sus características especiales y precio. Son tarjetas muy económicas, hasta tal punto que se pueden encontrar por 3$ en ciertas páginas web ofreciendo características similares incluso adicionales como wifi o bluetooth, formas de conexión inalámbricas con las que comunicarse con los dispositivos externos. Este modelo en concreto presenta hasta dos DAC, indispensables para llevar a cabo este proyecto y 16 entradas analógicas, número suficiente para la aplicación en concreto. Aparentemente sería la opción ideal por especificaciones y precio, pero su voltaje de alimentación (3,3v) y por lo tanto limitación para la salida de señales analógicas lo descarta por completo al igual que la tarjeta Arduino Due.

Una vez estudiado todas las posibilidades y decidida la placa a utilizar, se pueden comentar las siguientes características del Arduino UNO más relevantes a comentar para el diseño del prototipo posterior:

- Microcontrolador: ATmega328
- Voltaje de operación: 5V
- Voltaje de entrada (recomendado): 7-12V
- Voltaje de entrada (límites): 6-20V
- Pines de E/S digitales: 14 (de los cuales 6 proporcionan salida PWM)
- Pines de entrada analógica: 6
- Corriente DC por pin de E/S: 40 mA
- Corriente DC para 3.3V Pin: 50 mA
- Memoria Flash: 32 KB de los cuales 0.5 KB utilizados por el bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Velocidad de reloj: 16 MHz

Para completar esta información sobre los componentes hardware:

- El microcontrolador tiene el objetivo de ejecutar el programa que se haya escrito previamente. Es el que además limita las conexiones, dispone de los pines de entrada y salida que posteriormente se colocan para un mejor acceso en la placa que conforma el Arduino.
- Los pines de entrada digital funcionan con valores binarios, por lo que una entrada de 5V supondrá un 1 y una entrada nula un 0. Además, estos 14 pines pueden enviar o recibir como máximo 40 mA, y se pueden ampliar estos 14 con los otros 6 utilizando las entradas analógicas como digitales.
- Las entradas analógicas son 6, obtenidas a partir de un convertidor analógico/digital incorporado de 10 bits. Miden por defecto de 0V a 5V.

- Las salidas PWM se obtienen de los pines 3,5,6,9,10 y 11 de la placa, los cuales tiene esta posibilidad de proporcionar una señal modulada por ancho de pulso. Esta es una señal periódica cuadrada de amplitud 5V con la que se puede controlar el ciclo de trabajo para tener una señal casi continua, asemejándose a una señal analógica. Esta variación de la señal es resultado del valor medio de la señal al cambiar el ciclo de trabajo.

- La comunicación I2C está habilitada por la incorporación en la placa del bus I2c, lo cual dota de mayores posibilidades en cuanto a conexión de dispositivos y además varios en un único bus. Debido a su utilización en este proyecto se ha explicado detalladamente en el apartado 3.3 de esta memoria.

Además de estas características, es relevante comentar que posee un oscilador de cristal de 16MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reset. El software de la placa incluye un controlador para simular un ratón, un teclado y un puerto serie. Con todo esto solo es necesario conectarlo a un ordenador por USB para comunicarse y a la vez alimentarse.

La distribución de los pines, alimentación, botón de reset se puede diferenciar perfectamente en la siguiente figura:

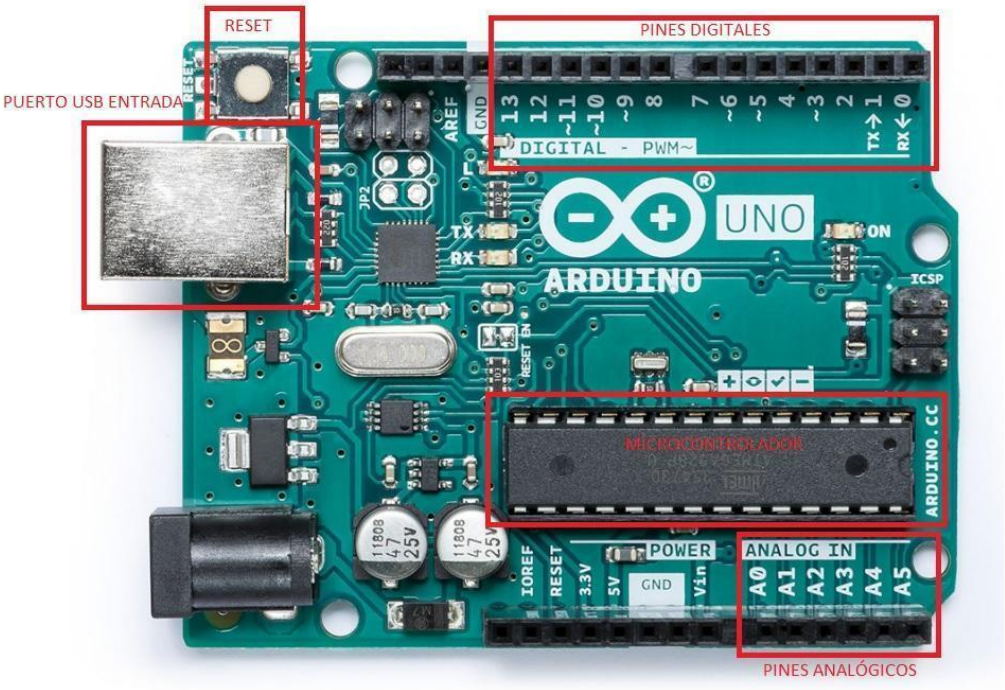


FIGURA 6: DISTRIBUCIÓN COMPONENTES ARDUINO UNO

3.3. COMUNICACIÓN I2C

I2C [6] es un protocolo de comunicación serial desarrollado por Philips Semiconductors en 1982 para la comunicación entre chips de sus televisores. Con él se define la trama de datos y conexiones para transferir bits entre dispositivos digitales. Además de protocolo también es un puerto compuesto de dos cables de comunicación, el cual permite conectar hasta 127 dispositivos en esas dos líneas con velocidades de hasta 1000kbits/s. Es uno de los más utilizados para la comunicación con sensores digitales con la gran ventaja de que su propia arquitectura incluya una confirmación de datos recibidos.

Dentro de las tarjetas Arduino, viene integrado dentro de dos puertos de entradas analógicas, diferentes para cada modelo. Con este protocolo se consigue conectar más sensorización y actuadores de los que en un principio limitan sus entradas y salidas.

Este protocolo viene instalado en un bus compuesto por dos hilos, entre los que se establece comunicación con dispositivos maestro/esclavo. Esto significa que existirán dispositivos que requieran un servicio, generen una petición y la coordinen (Maestro); y otros que se mantengan a la espera para responder a los requerimientos (Esclavo). Cada dispositivo conectado tendrá una dirección para que el maestro sepa a quién dirigirse y de quien recibir datos.

Dentro del bus que compone esta comunicación, existen dos líneas especificadas en todos los componentes compatibles:

- SCL: línea que lleva la señal de reloj.
- SDA: línea que lleva los datos.

Cada pulso en el pin SCL le indica al esclavo o receptor que lea el valor que se envía desde el pin SDA, estando un mensaje compuesto por una trama de dirección indicando a que esclavo va dirigido y uno o más tramas de datos. La condición de inicio la marca el pin SDA en estado bajo y el SCL en estado alto. La dirección del esclavo está compuesta de entre 7 y 10 bits, la cual tiene que ser única para cada esclavo en el bus. Cada mensaje presenta un bit llamado R/W que indica al maestro el tipo de operación a realizar, si este es 1 tiene que enviar información, si es 0 tiene que recibirla. En este mismo mensaje, para indicar el final de la trama de datos, existe un bit de reconocimiento, el cual debe de ser puesto a cero por el receptor indicando que fue recibido correctamente. Para finalizar la comunicación basta con poner ambos pines en estado alto, primero el SCL y luego el SDA para completar la condición de parada.

Este tipo de comunicación presenta varias ventajas frente a los puertos serial que también presenta Arduino, por ejemplo. Presenta la posibilidad de comunicarse con varios sensores o actuadores desde un mismo puerto, tiene mejor velocidad de transmisión (hasta los 5Mbps), consume menos pines del microprocesador que la comunicación SPI y permite la existencia de más de un maestro al contrario que esta.

La conexión con Arduino pasa por reconocer los pines SDA y SCL en la tarjeta. En el Arduino UNO son los A4 y A5 respectivamente, correspondientes a las entradas analógicas. Una vez identificados sólo hay que conectar cada dispositivo a estos pines. Desde el computador hay que declarar en el código la librería llamada Wire, la cual viene instalada por defecto en Arduino y es indispensable para interactuar con este bus de forma fácil y sencilla. Una vez desarrollado el código personalizado para la función a realizar, se puede utilizar este bus con todas las ventajas que ofrece.

4. OPC

4.1. INTRODUCCIÓN

En los últimos años, la tecnología OPC [7] ha tenido una amplia distribución en todo tipo de entornos industriales debido a su enorme facilidad de interoperabilidad e integración de sistemas. El acrónimo OPC significa Open Platform Communications, tecnología orientada al control de procesos auspiciada originalmente por Microsoft ante la imposibilidad de acceder a los datos recogidos de forma genérica. Hay que remontarse a finales del siglo XX para que un grupo de empresas con intereses comunes desarrollasen la primera versión de la especificación OPC.

Esta comunicación apareció ante las dificultades de integrar sistemas diferentes, por lo que un grupo de fabricantes se aliaron con Microsoft para buscar una solución a estos problemas. Contar con esta entidad les proporcionó la ventaja de disponer de un desarrollo rápido al construir el estándar sobre una tecnología ya existente como era COM. Computing Object Model (COM) fue una plataforma pionera en la comunicación entre procesos para su utilización en entornos distintos que posteriormente ha sido utilizada para otras muchas plataformas como en su día lo fue para OPC. La primera solución a este problema apareció a mediados de 1996, conformando posteriormente una fundación OPC para gestionar las diversas especificaciones de la que forman parte más de 300 miembros de todo el mundo.

FIGURA 7: LOGO FUNDACIÓN OPC – HTTPS://OPCFOUNDATION.ORG/

OPC no es tanto una tecnología sino una especificación técnica no propietaria, es decir, se ha redactado un estándar oficial sobre el funcionamiento y comportamiento de la tecnología sin un propietario detrás. Define un conjunto de modos de acceder a ciertos objetos informáticos, llamados interfaces estándar, basados en la tecnología OLE/COM de Microsoft con el objetivo de comunicar elementos software.

Por lo tanto, OPC se puede definir como un conjunto estándar de interfaces, propiedades y métodos para su uso en control de procesos y aplicaciones de automatización de la producción. Proporciona acceso general a tipos de datos simples, arrays y cadenas de datos, los cuales representan cualquier información que el servidor OPC desee exportar (datos de sensores, parámetros de control…). Este estándar no realiza procesamiento de datos, es una interfaz común, una ventana a los datos para comunicarse con dispositivos sin importar el tipo ni el software de control. Pretende facilitar la interoperabilidad entre aplicaciones de automatización como SCADAS, dispositivos y sistemas de campo como PLCs y aplicaciones de gestión informática (Excel), pero no está orientado precisamente a él ya que no posee todos los requisitos de un sistema en tiempo real ni garantías al respecto.

La arquitectura en la que se basa es de tipo cliente/servidor, por lo que será necesario que se configuren dispositivos como servidores OPC y que existan aplicaciones clientes de este mismo estándar que sepan conectarse a estos. Esta arquitectura consiste en un cliente que realiza peticiones al servidor para que este le dé respuesta. El servidor está en contacto con el sistema operativo para que este le asigne un puerto donde esperar a las solicitudes del cliente, el cual por su parte también debe solicitarlo para que el sistema operativo le asigne uno libre.

De esta manera se establece una comunicación en la que los servidores desarrollados pueden ser utilizados por cualquier software a medida (HMIs, SCADAs, controladores avanzados…) para intercambiar información.

FIGURA 8: ESQUEMA ARQUITECTURA CLIENTE/SERVIDOR

Lo ideal desde el punto de vista del integrador es que las aplicaciones software se comuniquen fácilmente con los dispositivos a nivel de campo y con otras aplicaciones. Antes de este estándar, este tipo de comunicación no era sencilla debido a que las interfaces no eran comunes, cada fabricante tenía la suya por lo que hacía imposible conectar dispositivos con distintos protocolos, limitando la conectividad. Como consecuencia de todo esto aumentaron los costes de

integración, mantenimiento y soporte, dio pie a la aparición de diversos drivers a medida para cada pareja de dispositivos surgiendo nuevos problemas al comunicar (inconsistencia de los mismos, características hardware no soportadas, conflictos…). Si un fabricante sacaba al mercado un nuevo dispositivo suponía la creación de numerosos drivers para la adaptación a otros componentes, algo que a día de hoy sigue pasando si no existe esta integración con el estándar OPC.

La solución por lo tanto sería esta, un estándar de comunicación que permita la comunicación sencilla entre aplicaciones y dispositivos de forma abierta y flexible para la adaptación a diversos ámbitos. Esto supone que las aplicaciones delegan en el sistema operativo para comunicarse con dispositivos, siendo esta comunicación independiente del dispositivo en cuestión.

Por lo tanto, pasamos de tener un driver específico para cada dispositivo o función que se quiera realizar, a tener un estándar común de comunicación que abarca todas la posibilidades y conecta directamente con el driver deseado.

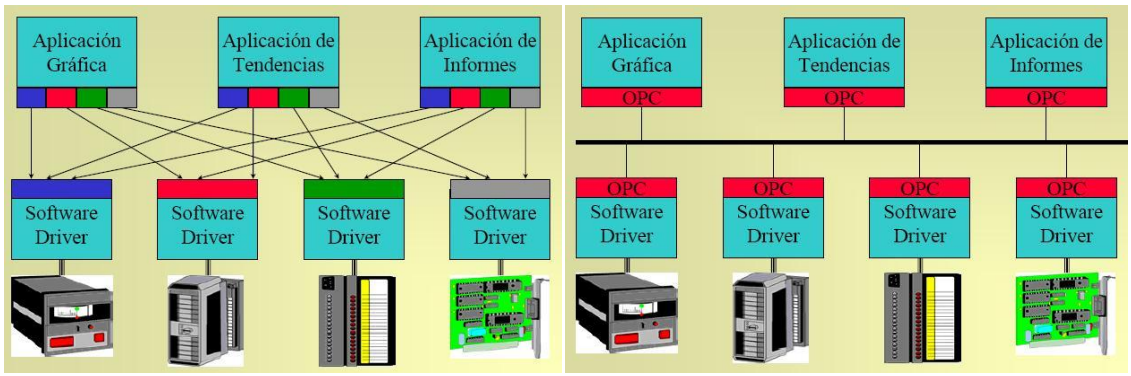


FIGURA 9 : COMPARATIVA COMUNICACIÓN CON OPC (DCHA) O SIN OPC (IZQ) – HTTPS://ES.WIKIPEDIA.ORG/WIKI/OPC

Por lo que vemos, OPC nos ofrece sendos beneficios principalmente por ser una tecnología abierta y estándar pero las principales características internas que lo definen son las siguientes:

- Ofrece un gran abanico de posibilidades en cuanto a la elección de un producto para una planta industrial, siempre y cuando lo soporte. Es una interfaz estándar por lo que el usuario se libra del compromiso con el propietario sin *ligarse* a un fabricante en concreto.

- Facilidad de sustitución. Es la llamada tecnología PnP (plug-and-play), el cual es cualquier avance que permite a un dispositivo conectarse a un ordenador sin tener que configurar controladores específicos. La sustitución por otro similar es inmediata ya que la interfaz se mantiene, facilitando enormemente la labor a los operarios (habrá que cambiar solo los identificadores).

- Presenta los datos de cualquier sistema de control de la misma forma, no hay múltiples protocolos o formas de conexión, todos funcionan de la misma forma.

- Acceso a los datos por parte de un mayor número de clientes, cualquier cliente OPC puede acceder a los datos del servidor, no es necesario caminos alternativos de conexión cuando se presenten diferentes esquemas de comunicación.

- Se elimina la obsolescencia al utilizar tecnologías como DCOM, extendiendo COM a redes inalámbricas para acceder a nuevas funcionalidades en objetos remotos. Se mantienen las interfaces existentes para permitir la compatibilidad con versiones anteriores.

- Los fabricantes obtienen ahorros de tiempo derivados de ofrecer sólo una única versión de su driver, consiguiendo la entrada a un mayor mercado lo cual facilita su conectividad e interoperabilidad. Habrá más clientes a los que les resultará más sencillo integrar su producto en los sistemas con este estándar de comunicación.

Uno de los que más se beneficia son los suministradores de plataformas HMI, Interfaz Hombre Máquina, los cuales hasta el momento tuvieron que invertir muchos recursos en desarrollar y mantener drivers propietarios para todas las redes industriales existentes. Los usuarios finales e integradores de sistemas consiguen las siguientes mejoras con respecto a tecnologías anteriores:

- Costes de integración bajos principalmente por el ahorro del tiempo que se consigue al interconectar sistemas. Anteriormente realizar la conexión de fabricantes diferentes era un problema con una gran inversión de tiempo necesario.

- Facilidad de conexión e interoperabilidad.

- Costes de aprendizaje reducidos ya que solo hay una forma de conexión estándar.

- Eliminación de bloqueos propietarios, lo que supone libertad de elección, mayor competencia y, por consiguiente, reducción de precios finales.

- Acceso a datos por cualquier jerarquía de automatización, los datos fluyen hacia aplicaciones de gestión con una forma de acceso más fácil.

En este caso en particular, para la adaptación al microcontrolador se utiliza un software llamado OPC Server for Arduino, desarrollado por 'Software Tools for markets'. Es una manera gratuita y sencilla de configurar un servidor OPC para permitir al usuario conectar su Arduino con cualquier sistema Scada OPC. Permite recibir y enviar información desde o hacia el software HMI Scada para desarrollar aplicaciones industriales gratuitamente. Solo es necesario instalar la librería OPC.h en los directorios de Arduino, desarrollar el código y aportar los datos de cada elemento que se conecte en tiempo real.

En los apartados siguientes, se entra más en detalle en la base y funcionamiento del estándar OPC de forma genérica, ya que la adaptación de 'st4markets' es llevar a cabo este protocolo de comunicación a Arduino en base a programación C++. Ya que el servidor no ha sido desarrollado, no es de interés para la memoria conocer los detalles y fundamentos específicos, pero sí sus objetivos y tipos de tecnología más utilizados.

4.2. Fundamentos OPC

El desarrollo del estándar vino motivado por una necesidad de disponer de mecanismos para comunicarse con fuentes de datos de dispositivos a nivel de campo en una planta industrial e incluso con las bases de datos presentes en las mismas salas de control. En un entorno donde la automatización esté integrada y presente en la industria de procesos, los diversos elementos que forman parte de este sistema industrial, desde el punto de vista de la Automática, pueden seguir una distribución jerárquica piramidal con 4 niveles ordenados por su importancia y elementos que lo forman [8]:

1. Nivel de campo: compuesto por los elementos que interactúan directamente con el proceso, toda la instrumentación de la planta, desde sensores a actuadores.

2. Nivel de dispositivo: formado por los elementos que recogen estas señales pertenecientes al nivel anterior para procesarlas levemente y establecer una política básica o lógica con la que actuar. Estos son los dispositivos entrada/salida, los cuales acondicionan las señales para actuar a partir de un PID.

3. Nivel de control: se implementan las estrategias de control y supervisión más avanzadas como controles en cascada, feedforward…

4. Nivel de gestión: nivel más alto de la pirámide, por lo tanto, supervisa y ordena todo tipo de gestión de producción, optimización económica…

FIGURA 10: PIRÁMIDE ELEMENTOS EN SISTEMA INDUSTRIAL AUTOMATIZADO

En los niveles superiores la frecuencia de comunicación es menor que en los niveles inferiores, igual que la cantidad de información intercambiada. Aquí los servidores OPC median entre los elementos del segundo nivel y tercero, entre los que generan los datos y quienes intentan acceder a ellos. Los clientes OPC entonces se situarían un nivel por encima, entre la 3º y 4º capa, donde se encuentra el destinatario, quien debe decidir en función de la gestión que se quiere hacer el tipo de control más adecuado.

El servidor entonces debe tener una programación determinada, pudiendo ser local, en el mismo PC que el usuario, o remoto, en un equipo diferente con acceso red (se utilizaría DCOM en vez de COM). Este servidor está compuesto por capas que se encargan de gestionar aspectos diversos del funcionamiento general, la capa más externa expone las interfaces diferentes a los clientes para que estos gestionen los objetos comunicando con el servidor. En la capa siguiente el servidor gestiona los objetos que han creado los clientes. La tercera se puede considerar como la caché, memoria interna del servidor para almacenar los valores más recientes y comunicarse con ella de una forma mucho más rápida. En la última capa estaría la lógica de control de acceso al dispositivo, con funciones de bajo nivel que se comunican con un driver del propio dispositivo para realizar operaciones de entrada y salida, resultando solo válido para un dispositivo en particular.

FIGURA 11: ESTRUCTURA SERVIDOR OPC

Nos queda el dispositivo físico, que puede ser tanto de control de procesos como uno de alto nivel como SCADA, por lo que los servidores OPC ofrecerían estos datos gestionados por tipo de paquete determinado. Es evidente que el servidor OPC es el intermediario entre el cliente OPC y el dispositivo, por lo que en operaciones de lectura estos datos los almacena el servidor para su distribución, pero en operaciones de escritura el servidor actualiza los datos del dispositivo ordenado por el cliente.

4.3. Tipos de servidores

Dentro de este estándar de comunicación, existen diferentes servidores que ofrecen a los clientes distintos tipos de funcionalidades y especificaciones. El primer tipo de servidor y además el más popular, el de mayor uso en la industria, se denominó acceso a datos, Data Access o DA. El resto sólo añaden funcionalidades añadidas a este, como gestión de alarmas, seguridad de acceso a datos, etc. Por último, la OPC UA trata de sustituir las anteriores haciendo uso de servicios web en vez de apoyarse en el ya conocido DCOM para solucionar los problemas de comunicación que este acarrea, prescindiendo además de un soporte propietario (Microsoft).

Hay multitud de especificaciones publicadas en la fundación OPC con las que poder conseguir una gran variedad de características para la comunicación OPC, desde alarmas hasta acceso a datos históricos. Las más usuales y que aportan un mayor valor por lo que se entrará más en detalle posteriormente, son las siguientes:

- *OPC DA (Acceso a datos)*: permite la lectura y escritura de datos entre aplicación y dispositivo de control de procesos, siempre y cuando este sea estándar. Posee numerosas versiones y cada cual incluye compatibilidad con las anteriores añadiendo las nuevas funcionalidades establecidas.

- *OPC UA*: se obtiene del objetivo de Microsoft de eliminar DCOM en favor de .NET, consiguiendo integrar las especificaciones anteriores con transportes SOAP y HTTP(S), junto con un mensaje binario codificado en TCP. Este tipo de servidor aprovecha las nuevas tecnologías en favor de la automatización industrial, permitiendo el intercambio de información y datos en todo el ámbito de la máquina. Es capaz de trabajar con cualquier sistema operativo, tiene una arquitectura orientada a servicios por lo que la hace más fácil de integrar en un dispositivo, sobre todo en los que se han ido desarrollando en los últimos años. Dado el gran número de dispositivos que forman una fábrica moderna, tanto de visión artificial como de bases de datos, es la forma más adecuada para conectar todas estas islas y centralizar la información.

Uno de los aspectos más difíciles de entender es el papel que juegan los objetos e interfaces dentro del protocolo estándar OPC. Los objetos son las entidades con propiedades propias sobre las que realizamos una serie de acciones a través de sus interfaces. El cliente y servidor OPC son dos objetos, pudiendo el cliente comunicarse con varios servidores simultáneamente aunque sean de fabricantes diferentes y estar situados en equipos distintos. Una posibilidad muy habitual también es que un servidor reciba peticiones de varios clientes, creando la posibilidad de comunicación varios a varios. El servidor dentro de la especificación DA puede estar compuesto de más objetos, que a su vez jerárquicamente dependan de otros objetos.

Un servidor de acceso a datos suele ser un programa ejecutable con determinado código desarrollado por cada fabricante, que determina los datos a los que tiene acceso, sus nombres e implementa detalles de cómo físicamente se obtienen esos datos. Cuando el cliente quiere acceder a los datos que un servidor ofrece, se crea un objeto del mismo nombre, el cual va a hacer la función de interlocutor para ese cliente. Este objeto posee la información del servidor y sirve de contenedor para objetos grupo.

El objeto grupo es una entidad lógica que permite la organización de elementos en base a criterios del cliente, gestionando información sobre él mismo (propiedades de interés para su funcionamiento) y ofreciendo mecanismos para contener y organizar ítems a través de sus interfaces. Tanto este objeto como el resto están creados y alojados en el equipo servidor. Existen dos grupos de objetos, los locales o privados, creados por el cliente para su única visibilidad; y grupos públicos, para compartir entre varios clientes conectados al servidor.

Dentro de cada grupo existen estos ítems o elementos, definidos por el cliente para representar conexiones a fuentes de datos dentro del servidor. Estos datos pueden provenir de sensores en tiempo real, parámetros de control o estados de subsistemas. No se puede acceder a estos a través de ningún tipo de interfaz, se debe hacer a través del grupo que lo engloba. Este ítem consta de tres partes, el

valor, dato obtenido; la calidad, la cual representa si ese dato es fiable (GOOD o BAD); y la marca de tiempo, instante de tiempo en el que se obtuvo el valor o en el que el servidor lo actualizó.

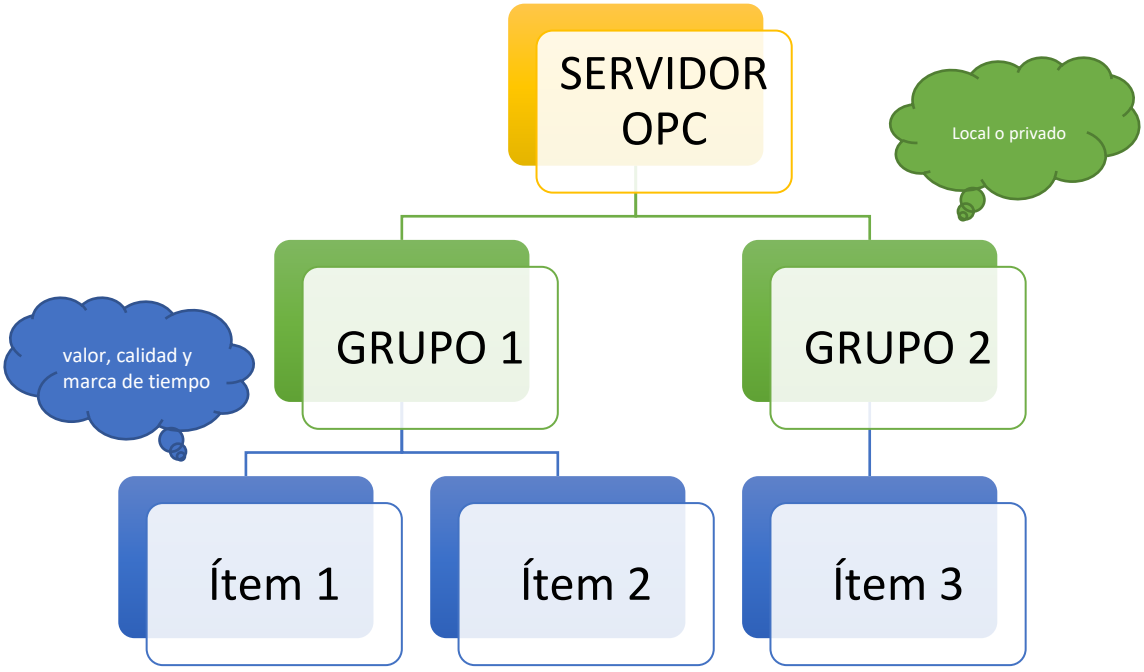


FIGURA 12: ESQUEMA ESTRUCTURA SERVIDOR OPC

Por lo tanto, para instalar un servidor OPC en un dispositivo determinado son necesarios 6 componentes esenciales.

1. *Sistema operativo*: el estándar OPC DA está basado en DCOM para un acceso distribuido a objetos, por lo que se necesita un sistema operativo que lo soporte. Esta es una tecnología de Microsoft, por lo que se necesita uno de esta compañía, desde Windows 98 hasta el más actual.

2. *Servidor OPC*: indispensable para acceder a los datos, debe estar instalado en un ordenador para acceder a un dispositivo en particular y ofrecer los datos según la especificación en cuestión DA. Este servidor puede ser suministrado por el fabricante o directamente programar uno, para lo que serán necesarias librerías para la comunicación OPC y saber acceder a los datos del dispositivo adquirido. En este último caso, la programación más habitual es en C++.

3. *Cliente OPC*: este nos permitirá acceder a los datos ofrecidos por el servidor a partir de un software como SCADA que lo soporte, configurándolo correctamente para su comunicación, o utilizar un entorno de programación simplificado como visual basic embebido en una hoja Excel, programando este acceso. Para esto vuelven a ser necesarias librerías cuyo uso viene determinado por la especificación OPC.

4. *Librerías dinámicas*: necesarias cuando se instala un servidor en un equipo determinado, se instalan automáticamente para facilitar las tareas de comunicación. La instalación se compone de un proxy de las interfaces comunes y otro de las de acceso a datos.

5. *Visualizador de servidores OPC*: cuando el cliente quiere conectarse a un servidor de una máquina remota, es necesario conocer los que están disponibles. El enumerador de servidores OPC para acceso remoto se distribuye con las librerías dinámicas capaz de acceder al gestor de categorías de componentes locales para suministrar este listado. Es un programa ejecutable y a la vez objeto de una interfaz para que los clientes puedan consultarle.

6. *Aplicaciones auxiliares*: en principio no es necesario nada más para el correcto funcionamiento, pero en ocasiones es necesario profundizar un poco más en aspectos de configuración cuando la comunicación remota da problemas. En estos casos se puede acceder al servicio de componentes de Windows para configurar propiedades de los objetos COM y la seguridad implementada en DCOM. Se puede acceder al registro de Windows para encontrar los servidores OPC y sus parámetros de registro asociados, como pueden ser el identificador de clase único, nombre o identificador de programa, localización…

Entrando más en detalle en las dos especificaciones con más características relevantes y más utilizadas en todo el ámbito industrial, se pueden destacar OPC DA y UA, que se tratará de explicar más en detalle en los próximos dos apartados.

4.4. OPC DA

En primera instancia se va a entrar en detalle en la especificación ya comentada OPC DA, Acceso a datos, para completar las pinceladas que se han dado.

Como ya se ha comentado, es la especificación fundamental para dar validez y funcionalidad a OPC, con la comunicación entre cliente y servidor. Está compuesta por 3 tipos de objetos con una relación jerárquica entre sí, siendo el servidor el pico de la representada pirámide, grupos la unidad central y el ítem el último elemento, con menos jerarquía pero más numeroso. Con el OPC DA se consigue el acceso a los datos, como su propio nombre indica, que ofrecen los dispositivos periféricos conectados cuyo objetivo es medir los fenómenos físicos en cada momento. La representación jerárquica, por lo tanto, trata de mostrar que un servidor depende de un gran número de grupos y que de cada grupo dependerá un gran número de ítems, siendo los dos primeros los únicos que tienen una interfaz con la que interactuar con el cliente OPC. Toda acción que se quiera realizar sobre los ítems debe de ser a través del objeto grupo que los contenga.

En cuanto a las interfaces, se puede entender que están asociadas a un objeto determinado que nos permite comunicarnos con él para pedir información por ejemplo o para dar una orden que modifique algún parámetro. Si el cliente solicita una conexión con un servidor, el sistema COM es el encargado de realizar las gestiones pertinentes para que el servidor se prepare, devolviendo un puntero a una interfaz a través de la cual el cliente pueda acceder al resto de interfaces ofrecidas por el servidor.

Como se ha visto hasta ahora, destaca que esta especificación tiene una funcionalidad principal, el intercambio de datos entre cliente y servidor. En este, el objeto servidor es importante: gestiona grupos y devuelve el estado, permitiendo el acceso fácil al espacio de nombres siendo la configuración que se haga de los grupos lo que determine cómo de buena será la comunicación. A la hora de crear un objeto grupo, son necesarios parámetros como el nombre, periodo de actualización, banda muerta, caché, asíncrono o flag entre otros.

Los ítems de OPC tienen unas propiedades asociadas que pueden ser accedidas a través de un interfaz, algo no habitual ya que el ítem en sí no tiene interfaz definida y por lo tanto no es posible realizar operaciones directamente sobre él. De todos modos, no suele ser algo frecuente, lo más usual es la lectura y escritura de valores y no acceder a una información adicional como rango, descripción o límites. Las propiedades pueden ser específicas, recomendadas o específicas del fabricante según el grado de obligatoriedad de las mismas. Algunas son de interés para establecer una serie de propiedades a los grupos, como la velocidad de acceso del servidor a los datos, la cual nos da una idea del valor mínimo que podemos fijar como periodo de actualización del grupo.

4.5. OPC UA

La arquitectura unificada OPC (Unified Architecture), la llamada OPC UA, es un protocolo independiente del proveedor y plataforma orientado a aplicaciones de automatización industrial. Se basa en el principio cliente servidor, orientada a una comunicación continua desde los sensores y actuadores presentes en una planta industrial hasta un sistema en la nube o ERP. Este ERP es el componente que gobierna todo, es un conjunto de aplicaciones de software que permiten automatizar la mayoría de las prácticas de negocio relacionadas con los aspectos operativos o productivos de una empresa, centralizando la información de todas las áreas que la componen, desde compras a producción o logística. Existen por supuesto otros protocolos de comunicación para la automatización industrial como PROFINET para PLC Siemens pero por ejemplo en este caso se trata de una tecnología específica con difícil integración en cualquier otro dispositivo.

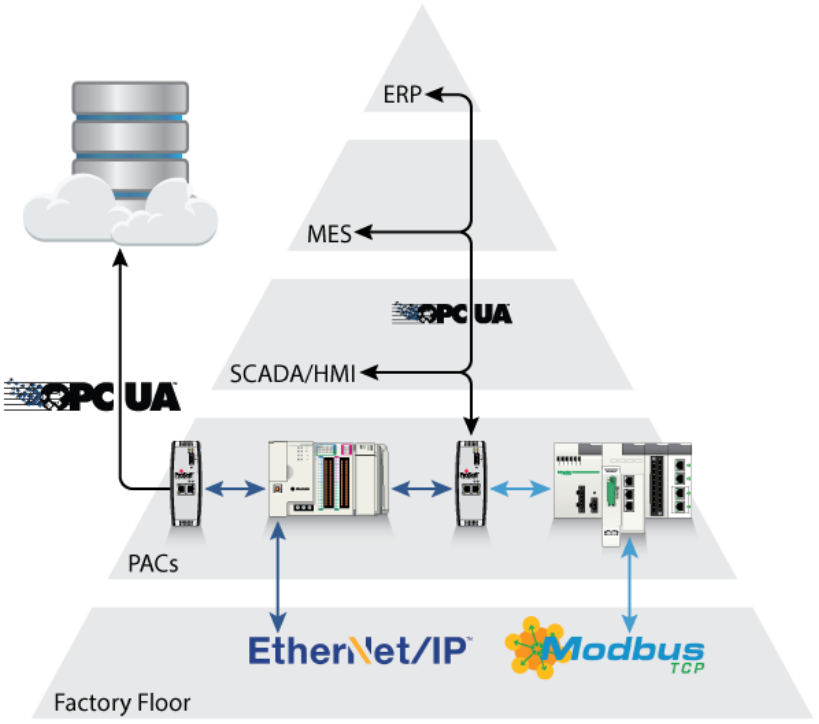


FIGURA 13: REPRESENTACIÓN JERÁRQUICA DE OPC UA – HTTPS://MX.PROSOFT-TECHNOLOGY.COM/LANDING-PAGES/OPC-UA

Se trata de un protocolo flexible e independiente, por lo que se le considera ideal para su implementación en la industria 4.0. Elimina la necesidad de utilizar los sistemas tradicionales de transferencia de datos como el bus de campo a pasar a transferirlos mediante un único protocolo dentro de la misma máquina de producción, entre máquinas o entre ella y la base de datos en la nube.

Es considerado en estos momentos el conjunto de protocolos de comunicación estándar entre dispositivos y sistemas industriales. En él se incorporan las grandes funcionalidades de los OPCs anteriores solucionando problemas de comunicación que requieren los actuales entornos industriales. La versión más utilizada hasta el momento, OPC DA, destacaba por su interoperabilidad y la no dependencia de ningún fabricante como se ha comentado, ahora bien, OPC UA incluye las siguientes mejoras sobre él que lo han hecho destacar en los sectores industriales de la actualidad:

- Fácil integración: OPC DA no ofrecía una gran comunicación entre clientes y servidores remotos, algo cada vez más demandado. OPC UA monta este tipo de comunicación de una forma rápida y sencilla, con menores tiempos de integración y costes.

- Fiabilidad de los datos: incorpora herramientas para dotar de una mayor seguridad a los datos haciendo imposible descifrar la comunicación entre

cliente y servidor. Este protocolo encripta los mensajes y solicita autenticación en las aplicaciones que quieren acceder a ellos.

- Interoperabilidad a todos los niveles: OPC DA ya permitía el intercambio de datos entre dispositivos y entornos industriales con independencia del fabricante en cuestión, pero OPC UA ya no está ligada a Windows, por lo que se puede utilizar con otros sistemas operativos, dispositivos IoT sencillos o herramientas cloud.

- Alcance de las arquitecturas en tiempo real: la suma de todas estas ventajas hace posible realizar grandes proyectos de recogida, explotación y análisis de datos en tiempo real, cuando anteriormente sólo era posible mediante intercambio de fichero históricos, sin la capacidad de actuar con datos a tiempo real.

- Múltiples opciones de mercado: al ser considerado un estándar y su gran adaptación a cualquier sistema y dispositivo, los software de gestión de datos industriales suelen incorporar driver de comunicaciones OPC UA, haciendo a los PLCs cada vez más capaces de recoger datos y transferirlos de esta manera.

Por lo tanto, OPC UA es una evolución de la tecnología clásica orientada a la comunicación industrial multi plataforma, abierta, segura y con ricos modelos de información. Está pensada para comunicar datos de equipos industriales con la principal diferencia de que no se limita a una comunicación entre aplicaciones y sensores sino que pueda comunicarse con todas las aplicaciones de la empresa y a través de todas las capas industriales. Con esto se consigue llevar los datos de las máquinas a donde se desee, de forma segura y multiplataforma (aplicaciones en el móvil, programas de gestión, hojas de Excel…).

5. DESARROLLO HARDWARE Y SOFTWARE

5.1. PROTOTIPO INICIAL

Una vez se han dado las nociones básicas de funcionamiento del protocolo de comunicación OPC y mostrado las posibilidades de desarrollo con una tarjeta Arduino, se puede mostrar la solución adoptada.

Recordando el objetivo, se debería poder sustituir una tarjeta de adquisición de datos por una placa oficial Arduino. El modelo elegido ha sido el Arduino UNO con las diferentes especificaciones que este ofrece, pero para llegar a ofrecer todas las prestaciones de la MCC en cuestión es necesario dotar a este hardware de unos dispositivos en concreto y de una programación que haga funcionar todo como un conjunto.

El primer punto a tratar es obtener una señal analógica pura para el control de los actuadores existentes en cuestión, en este caso concreto, una bomba de agua CC. Para ello hay que aportar la energía necesaria en forma de voltios para conseguir la velocidad deseada, en este caso entre 0 y 5V, siendo 0 V la bomba parada y 5V la bomba a máxima velocidad. Para conseguirlo se añade un dispositivo al Arduino con el que consigue aportar una señal analógica determinada en función de su voltaje de funcionamiento. Esta limitación hacía inviable la utilización de tarjetas como el Arduino DUE, que funcionan a 3.3V imposibilitando aportar la tensión de control necesaria a la bomba para aprovechar todo su rango de funcionamiento. Por lo tanto, el dispositivo elegido para esta función es el DAC MCP4725, con una resolución de 12 bits, el cual se comunicará con Arduino mediante el bus I2C ya comentado. Para aportar una mayor versatilidad, aprovechándose de las ventajas del bus I2C, en este mismo se colocará otro dispositivo MCP4725 para tener otra salida analógica real.

Por otra parte, ya se conocen las características y limitaciones de la placa Arduino UNO, la cual tiene 6 entradas analógicas con una resolución de 10 bits, y teniendo en cuenta que dos de estas están ocupadas para el bus I2C, quedan sólo 4 entradas analógicas y además con una resolución quizá escasa para determinadas aplicaciones donde sea necesaria una precisión más elevada. Como elemento auxiliar para solucionar estos problemas se ha optado por la adición del dispositivo ADC ADS115, el cual presenta cuatro conversores analógico digital ADC con una resolución de 16 bits en una placa conectada con Arduino a través también del bus I2C.

Estos dos tipos de dispositivos adicionales tendrán que controlarse a través de Arduino, por lo que además del código para nutrir al servidor OPC de datos será necesario añadir el correspondiente a las funciones de estos. Todo esto junto a la correspondiente conexión forman el prototipo para adquirir datos de los sensores y llevarlos a una computadora para tratarlos. Para optimizar esta conexión entre dispositivos, se ha desarrollado una placa de circuito impreso con el rutado correspondiente para facilitar el acceso a los diversos puertos y mejorar la fiabilidad. Por último, como añadido se ha diseñado y fabricado una caja personalizada para

esta PCB y placa Arduino donde encaja y se adapta perfectamente. Es necesario una cobertura para el prototipo y cualquiera que estuviese especializada para la placa de Arduino no tendría en cuenta esta PCB y la nueva ubicación de los puertos, por lo que se ha optado por diseñar una en 3D para fabricar posteriormente.

En los dos apartados siguientes, se entrará más en detalle en estos dos dispositivos adicionales a la tarjeta Arduino para conocer sus posibilidades, limitaciones y como se ha integrado en este proyecto.

5.2. DAC MCP 4825

Este módulo del fabricante Adafruit [9] permite generar una señal analógica a partir de una señal digital vía comunicación I2C. Presenta un pin de dirección accesible al usuario con el que poder elegir dos direcciones diferentes de comunicación dentro del bus y por lo tanto poder conectar dos módulos iguales al mismo puerto I2C. Para ello basta con soldar con estaño en esta parte del chip para comunicar el puente que hay en él y permitir el paso de corriente, lo cual internamente indica una dirección distinta a la anterior.

Esta señal se podría conseguir mediante las señales PWM, pero no son señales analógicas reales aunque se apliquen filtros como el RC para asemejarse a ellas [10].

En cuanto a características más técnicas y específicas:

- Alimentación: entre 2.7V y 5V (esta alimentación va a limitar la salida)
- CHIP: MCP4725
- Resolución: 12 bits
- Comunicación: Interfaz I2C
- Memoria EEPROM (mantiene el nivel de tensión aunque se corte la alimentación)
- Corriente máxima: 25mA
- Tiempo cambio salida: 6 µs
- Precio: 0.70€

El esquema para su montaje es muy sencillo, representado en la figura inferior.

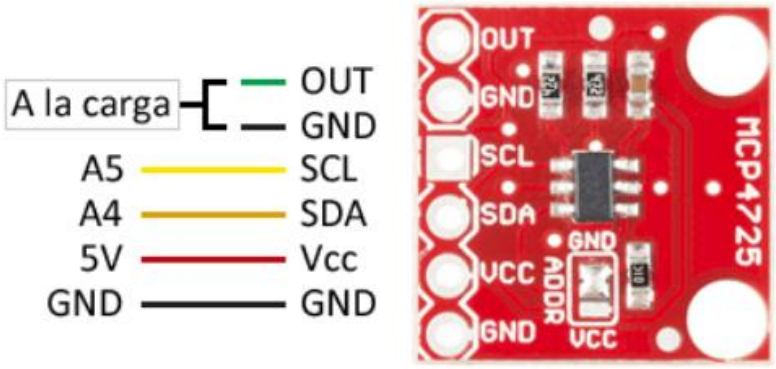


FIGURA 14: CONEXIÓN PINES MCP4725 CON ARDUINO

Para el correcto funcionamiento y ajuste a Arduino, es necesario incluir código que controle todos los componentes del mismo. Para ello, el propio fabricante Adafruit ofrece su librería "Adafruit_MCP4725.h" con la que sacar el máximo provecho de su producto. Una vez incluida, es necesario dar nombre a este dispositivo dentro de Arduino con lo que poder establecer el voltaje que se desee en cada momento.

A continuación, se explica las posibilidades que se ofrece de personalización a partir del código:

- *#include <Adafruit_MCP4725.h>* : función que incluye la librería para su utilización dentro de Arduino.

- *Adafruit_MCP4725 dac* : se le da un nombre al dispositivo, en este caso 'dac' para hacer referencia a él cuando exista otro dispositivo igual a este.

- *dac.begin(0x60)* : se establece una dirección para su comunicación a partir de I2C, en este caso la 0x60. Si no se indica nada por defecto aparecerá en la 0x62.

- *dac.setVoltage(4095, false)* : se establece conexión con el dispositivo nombrado como 'dac', para ofrecer una salida determinada a través de sus pines OUT-GND. En este caso se establece una salida de 4095, la cual es 5V teniendo en cuenta que la salida se establece por niveles según la resolución del dispositivo, en este caso 12 bits, por lo que 2^12= 4096 es número máximo de niveles que puede generar, desde 0 a 4095. El segundo término de la función setVoltage, en este caso false, representa un valor booleano para indicar si se quiere guardar esta salida en la memoria EEPROM para que en caso de que ocurra un corte en la alimentación se mantenga el valor establecido. Internamente, por defecto, incluye la frecuencia de escritura al dispositivo, en 400KHz.

5.3. ADC ADS1115

Este segundo dispositivo externo, también del fabricante Adafruit, dota al sistema de adquisición de datos en cuestión de 4 entradas con las que medir señales analógicas, todo esto mediante conversores analógico digitales como los que incorporan las placas de Arduino. Este dispositivo permite tener 4 entradas adicionales, las 4 restantes de Arduino con la ocupación de dos por el bus I2C podrían ser en ocasiones escasas. Además se consigue una mayor resolución en estas entradas, 16 bits en vez de 10.

Este ADS1115 tiene 16 bits de resolución, los cuales 15 son para la medición y 1 para el signo. Mediante el pin ADDRESS incorporado se pueden escoger entre 4 direcciones para la comunicación I2C dependiendo donde se realice la conexión, por lo que de la misma manera se podrán conectar hasta 4 dispositivos iguales. Otra ventaja de incorporar estas 4 entradas analógicas adicionales es que la conversión se realiza en el mismo dispositivo, por lo que se reduce la carga del procesador de Arduino. La medición se puede hacer también de forma diferencial utilizando dos pines contiguos, reduciendo así el ruido proveniente de utilizar una tierra común [11].

Una opción que incorpora pero que en este caso no se utiliza es la función de alerta, mediante la cual por el canal que lleva el mismo nombre se genera una señal cuando cualquiera de los canales sobrepasa un umbral establecido en el código. Permite además ajustar la ganancia dependiendo de las tensiones a medir, es decir, que si la tensión máxima a medir fuese menor a estos 5V máximos, se podría ajustar la ganancia para obtener precisiones superiores.

Las especificaciones técnicas más relevantes son las siguientes:

- Resolución: 16 bits
- Frecuencia de muestreo: 8 a 860 muestreos/segundo (programable)
- Alimentación: 2V a 5.5V
- Consumo de corriente: 150 µA (modo continuo)
- Direcciones I2C: 4 pines seleccionables

La conexión con la placa de Arduino es sencilla, es necesario conectarlo a la alimentación y los pines I2C conectando el pin de dirección a tierra por ejemplo para establecer una dirección concreta. Las otras tres direcciones serán conectando este pin a cualquiera de los otros 3 mencionados (SCL, SDA o GND). La figura inferior representa esta conexión.

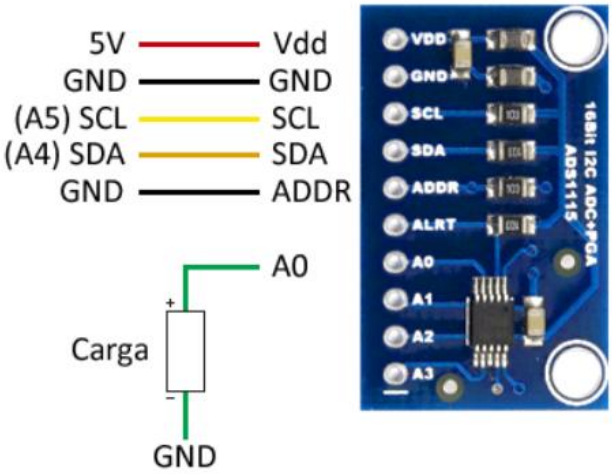


FIGURA 15: REPRESENTACIÓN CONEXIÓN ADS1115

De la misma forma que el otro dispositivo, es necesario incluir la librería correspondiente "Adafruit_ADS1015.h", nombrar a este dispositivo dentro del código y declarar la ganancia si se requiere.

A continuación, se explica las posibilidades que se ofrece de personalización a partir del código relacionado con la librería comentada:

- *#include <Adafruit_ADS1015.h>* : declaración de la librería con la que se cargan el resto de códigos.
- *Adafruit_ADS1115 ads :* nombre del dispositivo para referirse a él y asignarle una dirección concreta.
- *ads.setGain(GAIN_TWOTHIRDS)* : ajuste de ganancia para obtener más precisión. Este caso en concreto se puede omitir ya que es el que viene por defecto, para que 1 bit corresponda a 0.1875mV cuando la ganancia está entre +/- 6.144V. Si el argumento fuese GAIN ONE, estaría indicado para medir entre +/- 4.096V, GAIN_TWO entre +/- 2.048V, GAIN_FOUR entre +/- 1.024V, GAIN_EIGHT entre +/- 0.512V y GAIN_SIXTEEN entre +/- 0.256V.
- *ads.begin()* : se inicializa el dispositivo declarado como 'ads' en la dirección por defecto 0x48
- *int16_t adc0* : se declara la variable que almacenará la lectura por una de las entradas. Se deben declarar las otras 3 entradas de la misma forma con el nombre que se desee.
- *adc0 = ads.readADC_SingleEnded(0)* : se almacena en la variable declarada la lectura por la entrada del argumento, en este caso la A0. La entrada como se especifica en la función es en modo común, es decir compartiendo la misma tierra. De esta misma forma se procede a la lectura de las otras 3 entradas si se desea.

- *const float multiplier = 0.1875F* : la lectura de las entradas se hace en unidades, es decir, en este caso con esa ganancia y 15 bits tendríamos 2^15=32768 unidades, por lo que para pasarlo a voltios, hay que dividir la ganancia máxima 6.144V entre estas unidades, dando como resultado 0.0001875V/unidad.

- *int16_t results = ads.readADC_Differential_0_1()* : función de la librería para leer la entrada diferencial entre A0 y A1. Esta salida debe multiplicarse por el multiplicador de la misma forma que en el modo single end para obtener la salida en voltios.

- *ads.startComparator_SingleEnded(0, 13333)* : para funcionar en modo comparador, se utiliza esta función de la librería indicando como argumentos el pin de la entrada, A0 en este caso, que no debe de superar un valor, 13333 unidades.

- *adc0 = ads.getLastConversionResults()* : se declara la variable comparador de esa entrada con un nombre tipo int16_t y se actualiza el comparador.

5.4. ARDUINO OPC SERVER

El Arduino OPC Server es un software instalable en la computadora que configura a la placa de Arduino utilizada, según el modelo, en el servidor de la comunicación OPC a partir del código guardado en su interior. Cuando este se instala y configura, dentro del computador ya estará registrado como un servidor para la posibilidad de conexión y detección por parte de aplicaciones software clientes de OPC. Instalando el ejecutable desarrollado por "st4markets", se abre un programa con tres pestañas, dos de información y una para la configuración del servidor y su conexión con la placa de Arduino.

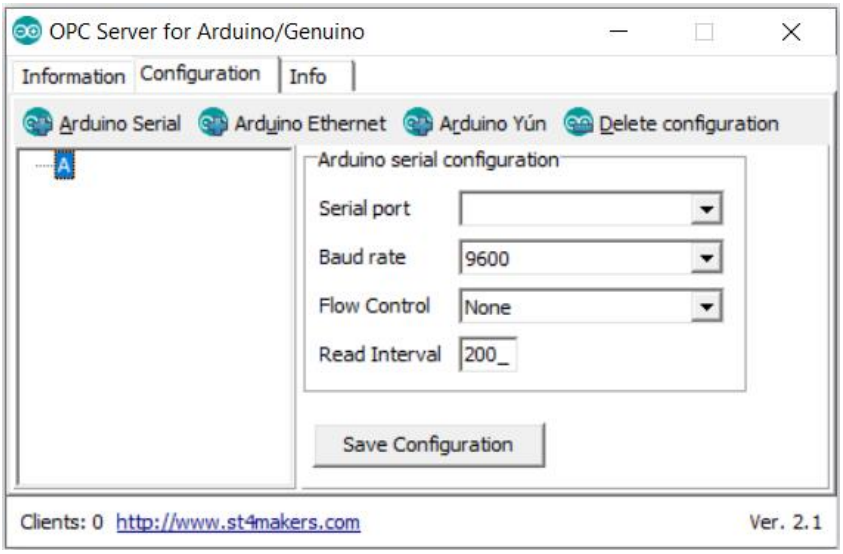


FIGURA 16: ASPECTO APLICACIÓN ARDUINO OPC SERVER

Dentro de esta, como se ha comentado, existe la posibilidad de concretar el modelo de Arduino en cuestión entre los que tengan comunicación Serial, Ethernet o con otros modelos Arduino Yún. Se selecciona el puerto en el que se establezca la conexión del Arduino por parte del computador, la velocidad de transmisión de datos en bits por segundo o lo que es lo mismo baudios, control de caudal, que no es necesario establecer y el intervalo de lectura, el cual se mantendrá por defecto en el valor de 200 micro segundos. Una vez configurado por completo, este programa se abrirá cada vez que se haga la comunicación OPC para actuar como servidor, permaneciendo en segundo plano.

Para utilizar la comunicación OPC en la placa de Arduino también es necesario incluir la librería correspondiente al software desarrollado por st4markets. La librería en cuestión es OPC.h, la cual tiene las siguientes funciones para sacar provecho a sus funcionalidades:

- *#include <OPC.h>* : se incluye la librería para el uso de OPC en Arduino y su comunicación.

- *OPCSerial aOPCSerial* : declaramos el objeto para la comunicación a través del bus Serial de Arduino con el servidor OPC y se le asigna un nombre.

- *int dac1(const char *itemID, const opcOperation opcOP, const int value)* : función clave de la comunicación. En esta se establecen las acciones a realizar cuando el servidor se comunica con un ítem para su lectura o escritura. Una vez dado un nombre indicativo a la función del ítem hacia el que va orientado, se tienen tres argumentos, siendo el primero el nombre del ítem que aparecerá en la aplicación del cliente, el segundo argumento el tipo de operación que va a establecerse en esta función y por lo tanto en el ítem (escritura, lectura o ambas) y por último el valor que se va a recoger del ítem.

- *aOPCSerial.setup()* : inicialización en la dirección de memoria por defecto de la comunicación con el servidor por el puerto serie.

- *aOPCSerial.addItem("OUT00_b",opc_readwrite, opc_int, dac1)* : relacionada con el tercer tipo de función de este servidor. Se toman los datos de esa para añadir el ítem al servidor, ya que ha sido completamente declarada antes y ahora sólo hay que especificar sus argumentos: nombre, tipo de comunicación, tipo de dato ofrecido y nombre de la función que recoge esta información.

- *aOPCSerial.processOPCCommands()* : establecida en la función loop() de arduino para que se lleve a cabo repetidamente. Esto se debe a que está enfocada a la actualización constante de todos los datos, busca los ítems que hayan sido actualizados tanto para leer como para escribir y los actualiza, realizando la comunicación entre el cliente y el servidor.

5.5. CÓDIGO ARDUINO

A continuación, se exponen las líneas de código a introducir en Arduino para su funcionamiento.

```
1.  #include <OPC.h>
2.  #include <Bridge.h>
3.  #include <Ethernet.h>
4.  #include <SPI.h>
5.  #include <Wire.h>
6.  #include <Adafruit_MCP4725.h>
7.  #include <Adafruit_ADS1X15.h>
```

En estas primeras siete líneas se incluyen las librerías adjuntas para el correcto funcionamiento de los dispositivos incluidos. Están las del MCP4725 Y ADS1115, la librería Wire para permitir la comunicación mediante I2C, Ethernet para controlar el Ethernet Shield que implementa la pila de protocolos TCP/IP de la comunicación OPC, la SPI para la comunicación cliente servidor de OPC y Bridge para facilitar la comunicación entre computadora y Arduino proporcionando funciones más simples.

```
8.  Adafruit_ADS1115 ads;
9.  Adafruit_MCP4725 DAC1,DAC2;
10. OPCSerial aOPCSerial;
```

En estas se declaran los nombres de los dispositivos a utilizar. Como existen dos MCP4725, a cada uno se le da un nombre para establecer con cada dispositivo una comunicación independiente a través de direcciones distintas pero en el mismo bus I2C. Al ADS1115 se le nombra como “ads” y al servidor OPC “OPCSerial” por la referencia a su método de comunicación.

```
11. static int dac1Value = 0;
12. static int dac2Value = 0;
13. const double multiplier = 0.1875F;
```

Se inicializan los valores de la variable que contendrá la salida de los DAC a cero para que cuando estos se desconecten partan de este valor. Se declara el multiplicador para el ADS1115 con la letra F al final del valor para indicar que es multiplicado por 10^-3.

```
14. int dac1(const char *itemID, const opcOperation opcOP, const int value){
15.   if (value>=0 && value<=4095){
16.     if (opcOP == opc_opwrite) {
17.       dac1Value = value;
18.       DAC1.setVoltage(dac1Value, false);
19.     }
```

```
20.      else
21.        return dac1Value;
22.    }
23.  }
```

En esta función int se está declarando la función que comunica desde el cliente la tensión que se quiere obtener a la salida del primer DAC MCP4725. Para ello hay que declarar una función con el nombre que se desee y tres argumentos definidos exactamente como aparecen ahí para su correcta interpretación a través de la librería. El primer argumento asigna un identificador al elemento, el segundo establece el tipo de comunicación que se va a realizar y el tercero el valor recogido. En este caso, el cliente establece una tensión con un valor determinado desde su aplicación, por lo tanto, la comunicación es de escritura de un valor determinado. Por ello, la condición para su escritura en el dac es que sea un valor válido, entre 0 y 4095, y que el tipo de operación sea de escritura. En este caso se establece el valor introducido por el cliente en el DAC para su salida. Si por el contrario el cliente no quiere escribir sino que quiere la lectura del mismo, cual es su salida, lo que será el resto del tiempo, esta condición última no se cumple y se devuelve el valor de la salida.

```
24.  int dac1_por(const char *itemID, const opcOperation opcOP, const int
     value){
25.    if (value>=0 && value<=100){
26.      if (opcOP == opc_opwrite) {
27.        dac1Value = value * 40.95;
28.        DAC1.setVoltage(dac1Value, false);
29.      }
30.      else
31.        return dac1Value/40.94;
32.    }
33.  }
34.  float dac1_v(const char *itemID, const opcOperation opcOP, const float
     value){
35.    if (value>=0 && value<=5){
36.      if (opcOP == opc_opwrite) {
37.        dac1Value = value*819 ;
38.        DAC1.setVoltage(dac1Value, false);
39.      }
40.      else
41.        return dac1Value/819.0;
42.    }
43.  }
```

De la misma manera si se quiere representar esta misma salida en diferentes unidades, como por ejemplo en este caso sobre un porcentaje o en bits. El cliente tendrá la posibilidad de observar y exigir la salida en la unidad más deseable en cada

momento. El cambio en este caso es el tipo de función que pasa a ser float para representar los voltios, consiguiendo así la posibilidad de representar decimales en los mismos.

```
44. int dac2(const char *itemID, const opcOperation opcOP, const int value){
45.   if (value>=0 && value<=4095){
46.     if (opcOP == opc_opwrite) {
47.       dac2Value = value;
48.       DAC2.setVoltage(dac2Value, false);
49.     }
50.     else
51.       return dac2Value;
52.   }
53. }
54. int dac2_por(const char *itemID, const opcOperation opcOP, const int
    value){
55.   if (value>=0 && value<=100){
56.     if (opcOP == opc_opwrite) {
57.       dac2Value = value * 40.95;
58.       DAC2.setVoltage(dac2Value, false);
59.     }
60.     else
61.       return dac2Value/40.94;
62.   }
63. }
64. float dac2_v(const char *itemID, const opcOperation opcOP, const float
    value){
65.   if (value>=0 && value<=5){
66.     if (opcOP == opc_opwrite) {
67.       dac2Value = value*819 ;
68.       DAC2.setVoltage(dac2Value, false);
69.     }
70.     else
71.       return dac2Value/819.0;
72.   }
73. }
```

Para el segundo convertidor digital analógico se ha realizado de la misma manera y con las mismas posibilidades como se observa en las líneas anteriores.

```
74. int a0(const char *itemID, const opcOperation opcOP, const int value){
75.   return analogRead(A0);
76. }
77. float a0_v(const char *itemID, const opcOperation opcOP, const float
    value){
78.   return (analogRead(A0) * 5.0 )/ 1024.0;
```

```
79.  }
80.  int a0_p(const char *itemID, const opcOperation opcOP, const int value){
81.    return analogRead(A0)/10.24;
82.  }
```

Estas líneas representan la lectura de un valor por las entradas analógicas de la placa arduino, más concretamente por la A0. El código se simplifica, partiendo de la misma función que para la lectura y escritura de los dos dispositivos DAC, para las entradas solo existe la posibilidad de lectura, por lo que cada función devolverá la señal en la unidad correspondiente, en bit, voltios y porcentaje respectivamente. Simplemente a la lectura se le aplican factores para pasar del bit en el que se ofrece la lectura por la función analogRead, a voltios o porcentaje, siendo el 100% los 5V máximos. Esto se realiza para el resto de entradas analógicas de la misma manera que en esta ocasión.

```
83.  int a1(const char *itemID, const opcOperation opcOP, const int value){
84.    return analogRead(A1);
85.  }
86.  float a1_v(const char *itemID, const opcOperation opcOP, const float
     value){
87.    return (analogRead(A1) * 5.0 )/ 1024.0;
88.  }
89.  int a1_p(const char *itemID, const opcOperation opcOP, const int value){
90.    return analogRead(A1)/10.24;
91.  }
92.  int a2(const char *itemID, const opcOperation opcOP, const int value){
93.    return analogRead(A2);
94.  }
95.  float a2_v(const char *itemID, const opcOperation opcOP, const float
     value){
96.    return (analogRead(A2) * 5.0 )/ 1024.0;
97.  }
98.  int a2_p(const char *itemID, const opcOperation opcOP, const int value){
99.    return analogRead(A2)/10.24;
100. }
101. int a3(const char *itemID, const opcOperation opcOP, const int value){
102.   return analogRead(A3);
103. }
104. float a3_v(const char *itemID, const opcOperation opcOP, const float
     value){
105.   return (analogRead(A3) * 5.0 )/ 1024.0;
106. }
107. int a3_p(const char *itemID, const opcOperation opcOP, const int value){
108.   return analogRead(A3)/10.24;
109. }
```

En las líneas siguientes la función de lectura cambia por la sencilla razón de que no existen más entradas analógicas disponibles en la tarjeta de Arduino UNO, por lo que se comienza a leer las que ofrecen el dispositivo conversor analógico digital ADC1115. La codificación por lo tanto será A4 para la entrada A0 del dispositivo, que aparece con el argumento 0 en la función, A5 para la A1, A6 para la A2 y A7 para la A3. De la misma forma que las anteriores entradas y salidas, se ofrece esta lectura en voltios y porcentaje, duplicando estas funciones, para una mayor versatilidad en el uso convencional.

```
110. int a4(const char *itemID, const opcOperation opcOP, const int value){
111.   return ads.readADC_SingleEnded(0);
112. }
113. float a4_v(const char *itemID, const opcOperation opcOP, const float
     value){
114.   return ads.readADC_SingleEnded(0)*multiplier/1000.000;
115. }
116. int a4_p(const char *itemID, const opcOperation opcOP, const int value){
117.   return ads.readADC_SingleEnded(0)*multiplier*0.02;
118. }
119. int a5(const char *itemID, const opcOperation opcOP, const int value){
120.   return ads.readADC_SingleEnded(1);
121. }
122. float a5_v(const char *itemID, const opcOperation opcOP, const float
     value){
123.   return ads.readADC_SingleEnded(1)*multiplier/1000.000;
124. }
125. int a5_p(const char *itemID, const opcOperation opcOP, const int value){
126.   return ads.readADC_SingleEnded(1)*multiplier*0.02;
127. }
128. int a6(const char *itemID, const opcOperation opcOP, const int value){
129.   return ads.readADC_SingleEnded(2);
130. }
131. float a6_v(const char *itemID, const opcOperation opcOP, const float
     value){
132.   return ads.readADC_SingleEnded(2)*multiplier/1000.000;
133. }
134. int a6_p(const char *itemID, const opcOperation opcOP, const int value){
135.   return ads.readADC_SingleEnded(2)*multiplier*0.02;
136. }
137. int a7(const char *itemID, const opcOperation opcOP, const int value){
138.   return ads.readADC_SingleEnded(3);
139. }
140. float a7_v(const char *itemID, const opcOperation opcOP, const float
     value){
141.   return ads.readADC_SingleEnded(3)*multiplier/1000.000;
```

```
142. }
143. int a7_p(const char *itemID, const opcOperation opcOP,       const int
     value){
144.   return ads.readADC_SingleEnded(3)*multiplier*0.02;
145. }
```

Una vez terminada la lectura de entradas y salidas junto a su escritura, se pasa a la función setup() que incluye todos los códigos de las tarjetas Arduino. Esta función supone que todo lo que se introduzca dentro de ella se va a ejecutar sólo y exclusivamente cuando se inicie el Arduino. Por el contrario la función loop() se ejecutará repetitivamente de arriba a abajo del código que la contenga mientras esté la tarjeta en funcionamiento y alimentada.

```
146. void setup() {
147.   Serial.begin(9600);
148.   DAC1.begin(0x60);
149.   DAC1.setVoltage(0, false);
150.   DAC2.begin(0x61);
151.   DAC2.setVoltage(0, false);
152.   ads.begin(0x48);
```

Para comenzar con esta función, se inicializan las direcciones de los dispositivos, se resetean los DACs a voltaje cero para evitar perturbaciones anteriores y que el valor que se desee establecer parta de cero en caso de desconexión de la tarjeta. Si la aplicación fuese diferente y en caso de desconexión se quisiese que este voltaje empezase en el último valor requerido, sería tan fácil como eliminar estas dos líneas de código. En el caso del puerto Serial por defecto con los 9600 de su argumento se establece la velocidad de comunicación en bits por segundo para la correcta lectura por parte del computador.

```
153.   aOPCSerial.setup();
154.   aOPCSerial.addItem("OUT00_b",opc_readwrite, opc_int, dac1);
155.   aOPCSerial.addItem("OUT00_%",opc_readwrite, opc_int, dac1_por);
156.   aOPCSerial.addItem("OUT00_V",opc_readwrite, opc_float, dac1_v);
157.   aOPCSerial.addItem("OUT01_b",opc_readwrite, opc_int, dac2);
158.   aOPCSerial.addItem("OUT01_%",opc_readwrite, opc_int, dac2_por);
159.   aOPCSerial.addItem("OUT01_V",opc_readwrite, opc_float, dac2_v);
160.   aOPCSerial.addItem("IN00_b",opc_read, opc_int, a3);
161.   aOPCSerial.addItem("IN00_V",opc_read, opc_float, a3_v);
162.   aOPCSerial.addItem("IN00_%",opc_read, opc_int, a3_p);
163.   aOPCSerial.addItem("IN01_b",opc_read, opc_int, a2);
164.   aOPCSerial.addItem("IN01_V",opc_read, opc_float, a2_v);
165.   aOPCSerial.addItem("IN01_%",opc_read, opc_int, a2_p);
166.   aOPCSerial.addItem("IN02_b",opc_read, opc_int, a1);
167.   aOPCSerial.addItem("IN02_V",opc_read, opc_float, a1_v);
168.   aOPCSerial.addItem("IN02_%",opc_read, opc_int, a1_p);
```

```
169.    aOPCSerial.addItem("IN03_b",opc_read, opc_int, a0);
170.    aOPCSerial.addItem("IN03_V",opc_read, opc_float, a0_v);
171.    aOPCSerial.addItem("IN03_%",opc_read, opc_int, a0_p);
172.    aOPCSerial.addItem("IN04+_b",opc_read, opc_int, a4);
173.    aOPCSerial.addItem("IN04+_V",opc_read, opc_float, a4_v);
174.    aOPCSerial.addItem("IN04+_%",opc_read, opc_int, a4_p);
175.    aOPCSerial.addItem("IN05+_b",opc_read, opc_int, a5);
176.    aOPCSerial.addItem("IN05+_V",opc_read, opc_float, a5_v);
177.    aOPCSerial.addItem("IN05+_%",opc_read, opc_int, a5_p);
178.    aOPCSerial.addItem("IN06+_b",opc_read, opc_int, a6);
179.    aOPCSerial.addItem("IN06+_V",opc_read, opc_float, a6_v);
180.    aOPCSerial.addItem("IN06+_%",opc_read, opc_int, a6_p);
181.    aOPCSerial.addItem("IN07+_b ",opc_read, opc_int, a7);
182.    aOPCSerial.addItem("IN07+_V",opc_read, opc_float, a7_v);
183.    aOPCSerial.addItem("IN07+_%",opc_read, opc_int, a7_p);
184.  }
```

En esta parte del código se añaden los ítems o elementos al servidor mediante la función “aOPCSerial.additem”. En esta, tal y como se ha establecido en su lectura y escritura, hay que añadirle un nombre único que lo identifique, el tipo de operación que se va a realizar, en este caso lectura y escritura con “opc_readwrite” para los DAC y lectura sólo para las entradas analógicas “opc_read”. A continuación se establece en el siguiente argumento el tipo de dato que será el ítem y por último el nombre que se le ha dado a la función que escribe o devuelve el dato. Como se observa, se ha declarado a las entradas analógicas del ADC1115 con un signo de “ + ” para indicar que son las de mayor resolución, la unidad de bit con una “ b ” al final del guión bajo y el nombre de la entrada, “ V ” para voltios y “ % ” para porcentaje sobre el total.

```
185. void loop() {
186.    aOPCSerial.processOPCCommands();
187. }
```

Para terminar, en esta función loop() repetitiva, sólo es necesario establecer esa función de la librería OPC para poder enviar y recibir datos desde o hacia el puerto serie como se ha explicado en el apartado anterior, sin esta no sería posible la comunicación.

Estos ítems, por lo tanto, en la aplicación por defecto utilizada para la comunicación OPC (Softing OPC Toolbox Demo [12]) aparecerán como en la figura inferior, superponiendo delante de cada nombre la nomenclatura que se le quiera dar al dispositivo Arduino, en este caso “A” para acortar los nombres, diferenciando estos ítems de otros que provengan de otra fuente.

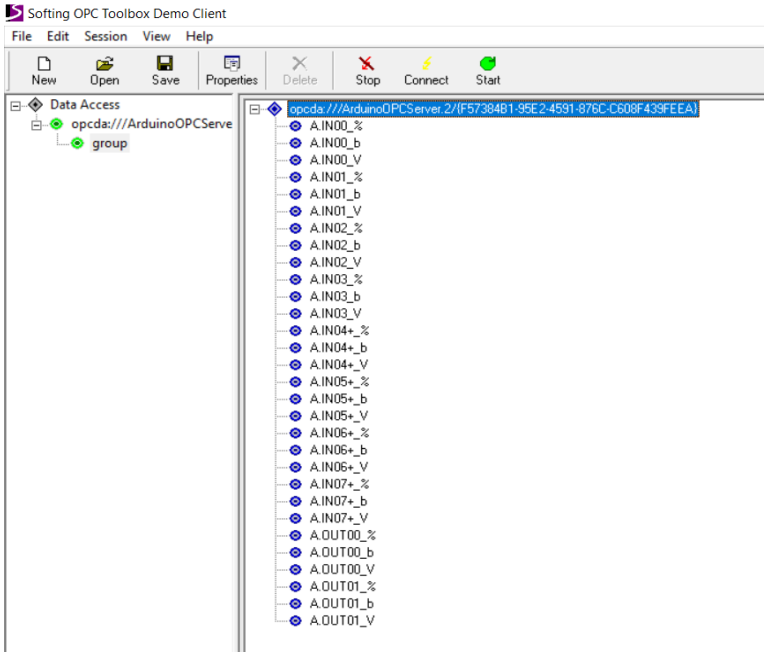


FIGURA 17: ESPACIO DE NOMBRES OPC EN SOFTING OPC TOOLBOX

5.6. DESARROLLO PLACA CIRCUITO IMPRESO

Para la conexión de los dispositivos adicionales comentados se procede a la realización de una placa de circuito impreso (PCB) donde establecer todas las conexiones, para lo cual se utiliza el software Microsim PSpice 8 [13] ofrecido por la escuela. Hay que tener en cuenta la ubicación de los terminales de la placa Arduino a utilizar para hacer su conexión con la PCB, la cual se colocará encima de la anterior y por lo tanto las salidas y entradas definitivas partirán de esta. Teniendo en cuenta estos condicionantes y las limitaciones de tamaño de la placa, se realiza un estudio de las posibilidades y se decide optar por la siguiente disposición.

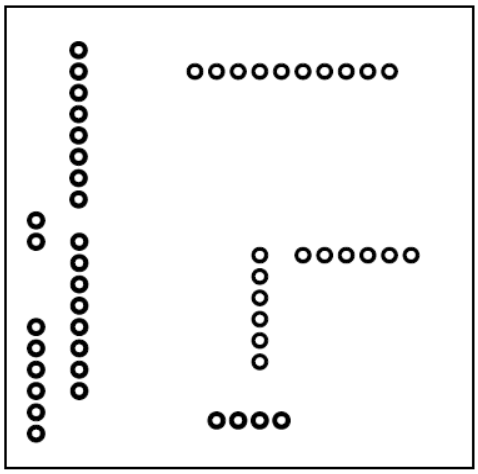


FIGURA 18: DISTRIBUCIÓN DE ORIFICIOS DE LA PCB

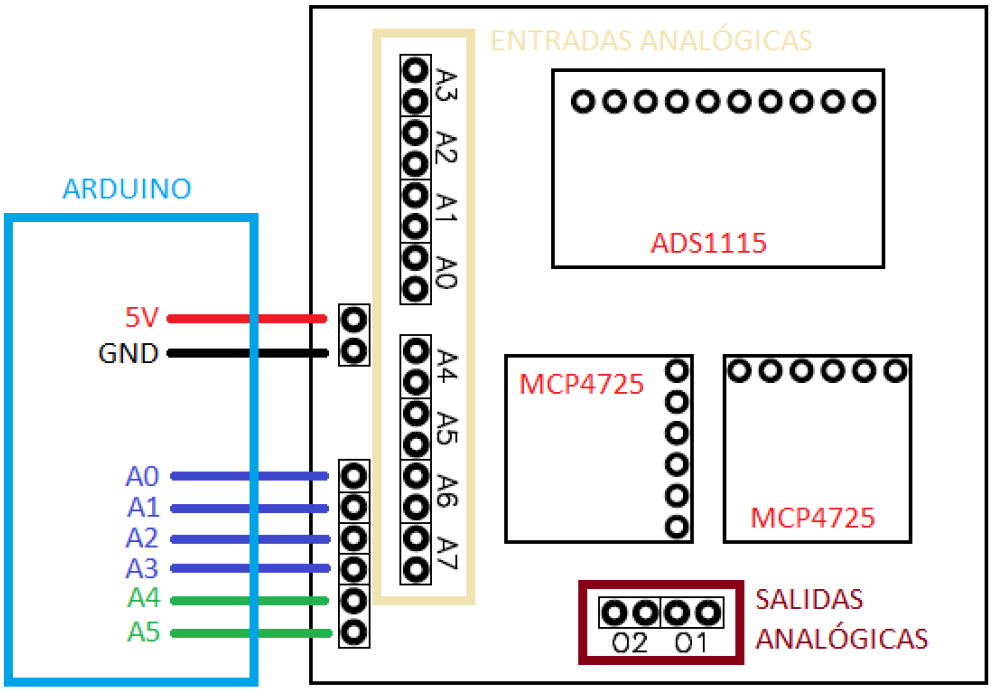


FIGURA 19: DISTRIBUCIÓN Y CONEXIÓN DE DISPOSITIVOS

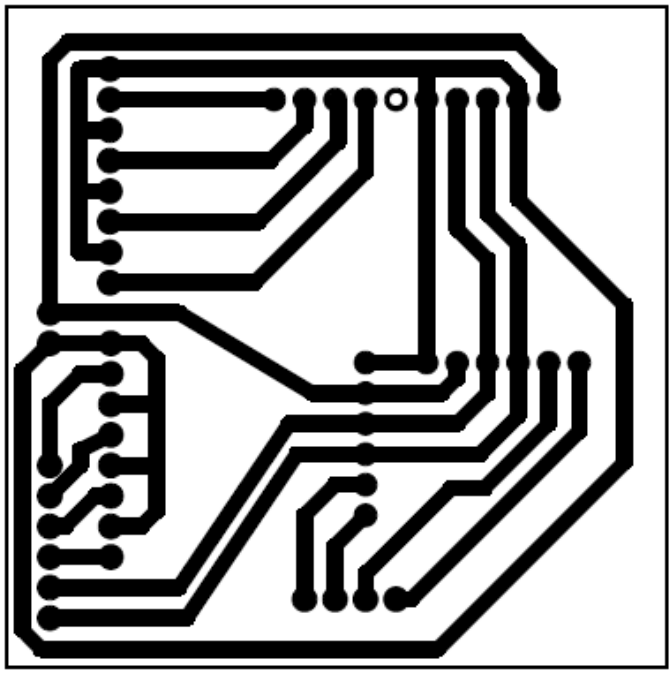


FIGURA 20: RUTADO PCB

Como se ha comentado, la disposición de los componentes se ha visto limitada por favorecer al rutado de estas pistas y la conexión de todos los componentes, llegando a una PCB bien diseñada que agrupa todas las entradas y salidas para un mejor acceso para el usuario final.

Una vez comprobada la viabilidad y estudiadas todas las posibilidades de distribución, se puede proceder a la fabricación de la placa. Para ello existe la posibilidad de hacerlo en el departamento de Electrónica de la Escuela, la cual posee los elementos necesarios para el proceso y además puede asesorarte en la configuración más adecuada para la función a cumplir.

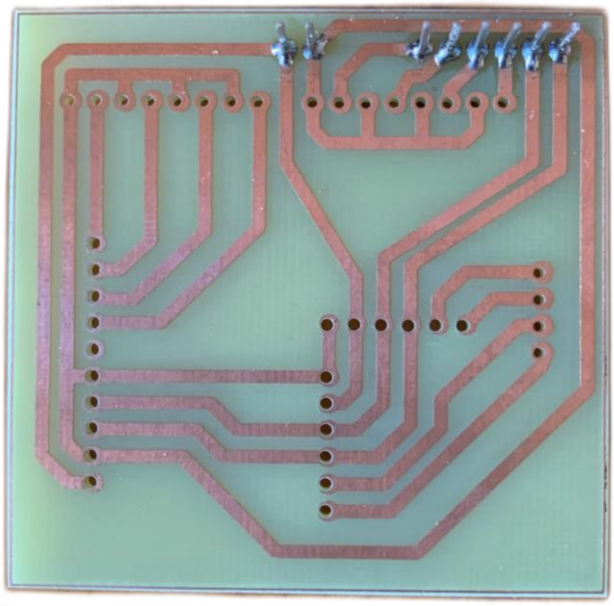


FIGURA 21: RUTADO PCB FABRICADA

A la placa una vez fabricada hay que soldarle los módulos adicionales, desde los dispositivos protagonistas hasta terminales para la entrada de cableado exterior y conexionado con la placa Arduino. Estos se instalan para facilitar la conexión con la instrumentación desde el exterior sin necesidad de tener a mano todos los terminales, solo los que interesen al prototipo final.

FIGURA 22: PCB COMPLETA CON DISPOSITIVOS Y TERMINALES SOLDADOS

Como se observa, los terminales verdes se sueldan para poseer una entrada de cableado desde el exterior sin que el usuario final tenga acceso directo al prototipo, por lo que la caja protectora de todo este dispositivo debe tener a estos en cuenta para sobresalir de la misma y poder acceder a ellos desde el exterior.

5.7. DESARROLLO CAJA ENVOLTORIO

Para todo el conjunto, tanto la placa de Arduino como la PCB fabricada, es necesario un envoltorio que lo proteja del exterior y lo englobe en un conjunto más práctico y manejable. La primera opción era una caja estándar para las placas Arduino, con los problemas evidentes de adaptación al prototipo fabricado, tanto en las entradas como salidas que cambian su disposición respecto al original o simplemente que no se utilizan. Teniendo claro que lo mejor es una adaptación al prototipo fabricado, otra opción es hacer las aperturas a mano en una caja estándar, pero existe la posibilidad de realizar su diseño personalizado para la posterior impresión en 3D. Con esta opción se hace un envoltorio al prototipo lo más adaptado a las prestaciones posibles sacando el máximo partido al mismo. En este caso se ha utilizado el software 'Autodesk Fusion 360' [14] para el diseño de las piezas y el 'Ultimaker Cura' [15] para la obtención de los códigos necesarios para la impresora 3D, modelo 'Ender 3 Pro'.

Se ha dividido este diseño en dos partes, la inferior y la superior que posteriormente se unen con tornillería. Las medidas completas de todo el envoltorio se establecerán en los anejos; en esta memoria se presenta el resultado final.

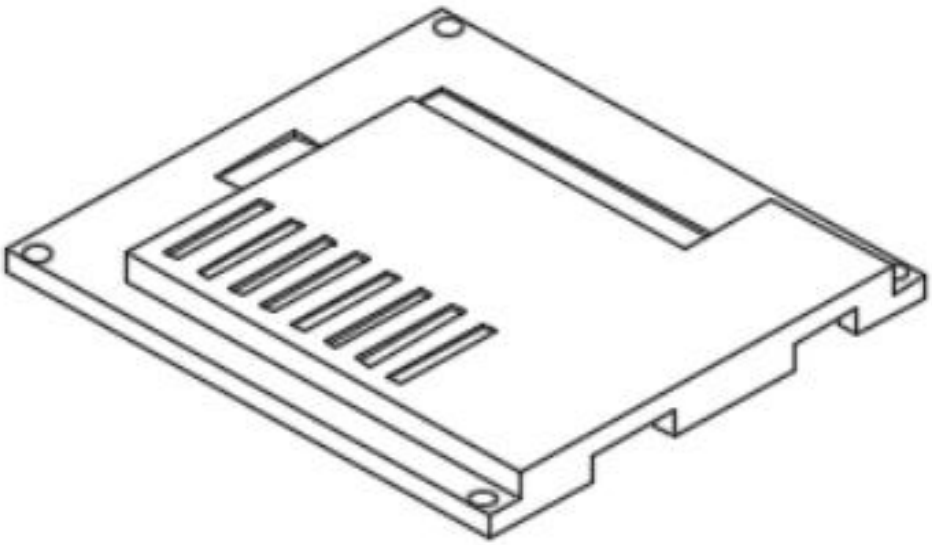


FIGURA 23: PROTOTIPO PARTE SUPERIOR 3D VISTA SUPERIOR DELANTERA

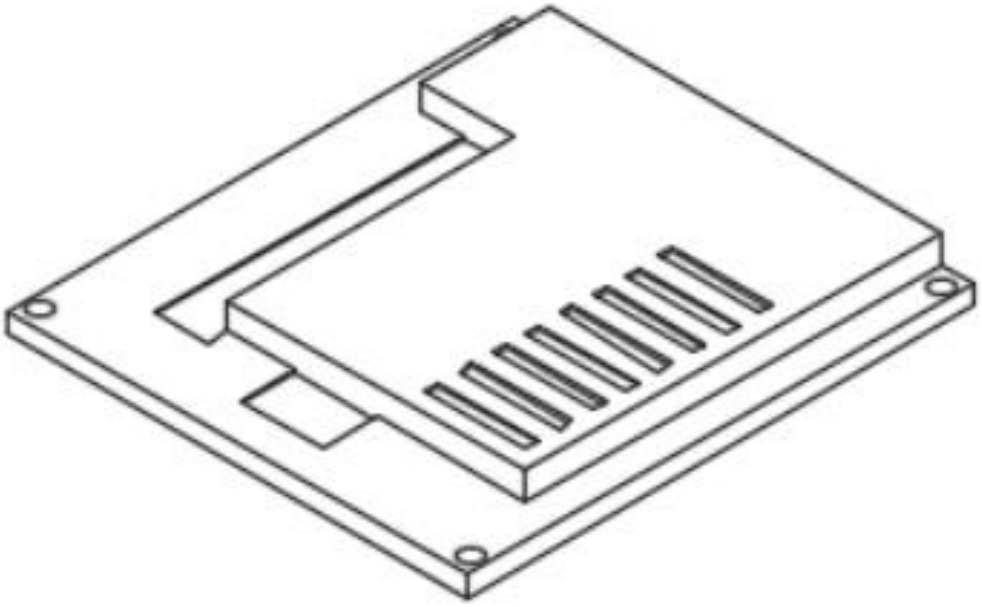


FIGURA 24: PROTOTIPO PARTE SUPERIOR 3D VISTA SUPERIOR TRASERA

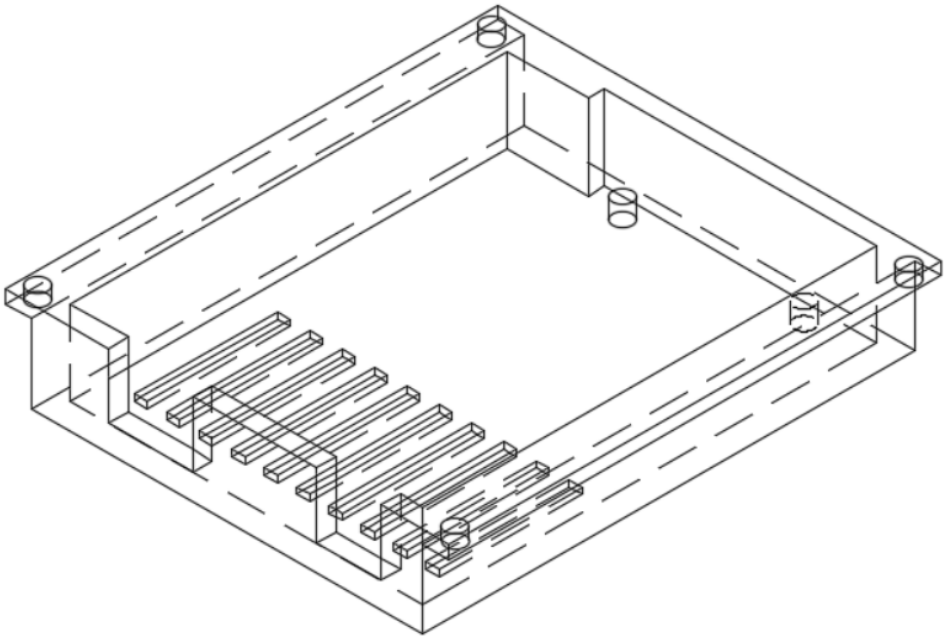


FIGURA 25: PROTOTIPO PARTE INFERIOR 3D VISTA SUPERIOR DELANTERA

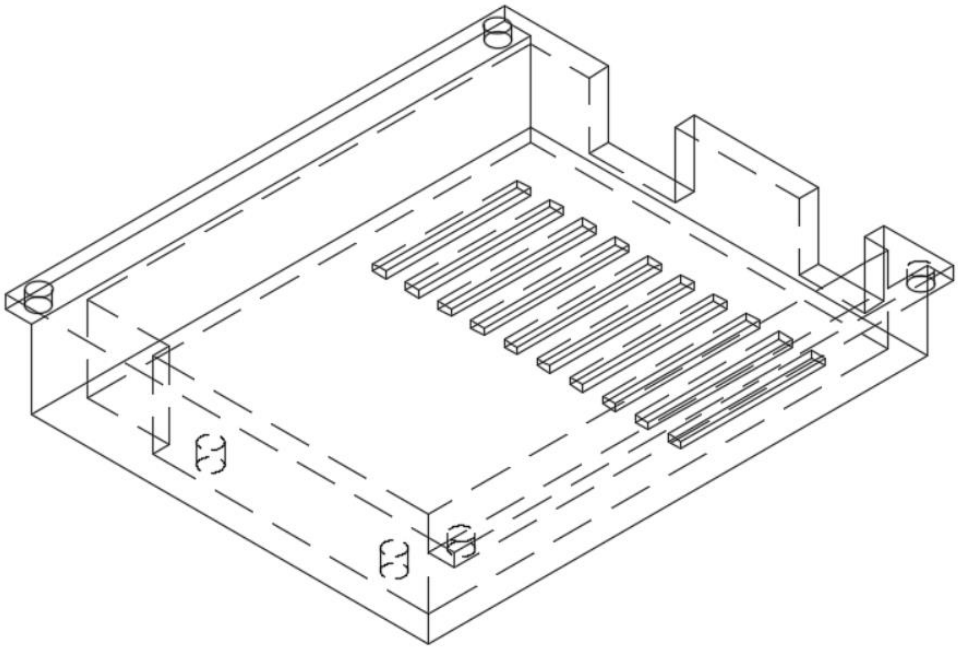


FIGURA 26: PROTOTIPO PARTE INFERIOR 3D VISTA SUPERIOR TRASERA

El resultado a partir de la impresora 3D y el diseño tomado, tras hacer las comprobaciones pertinentes de ajuste y medida es el mostrado en las figuras 27 y 28.

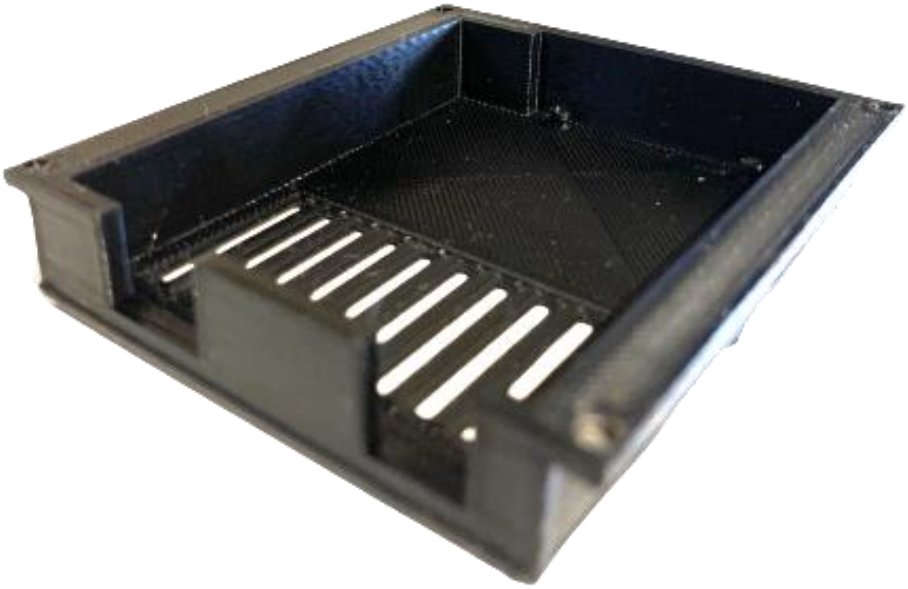


FIGURA 27: PROTOTIPO FABRICADO PARTE INFERIOR

FIGURA 28: PROTOTIPO FABRICADO PARTE SUPERIOR

El resultado como se puede observar es el esperado, la caja se ajusta perfectamente al prototipo, presenta rejillas tanto en la parte superior como en la inferior para favorecer su ventilación evitando el calentamiento de componentes, tiene dos aperturas en su parte superior para la salida de los terminales de entrada y salida y la unión entre ambas partes se realiza con 4 tornillos de 2mm rosca chapa que unen perfectamente ambas piezas.

5.8. PROTOTIPO FINAL

Para concluir y de forma general, junto con la placa y el envoltorio, se tiene un prototipo final personalizado, compacto y adaptado a las prestaciones que el mismo ofrece. Una vez introducidos los terminales de la PCB necesarios para las conexiones, la unión entre ambas placas se presenta como la figura inferior.

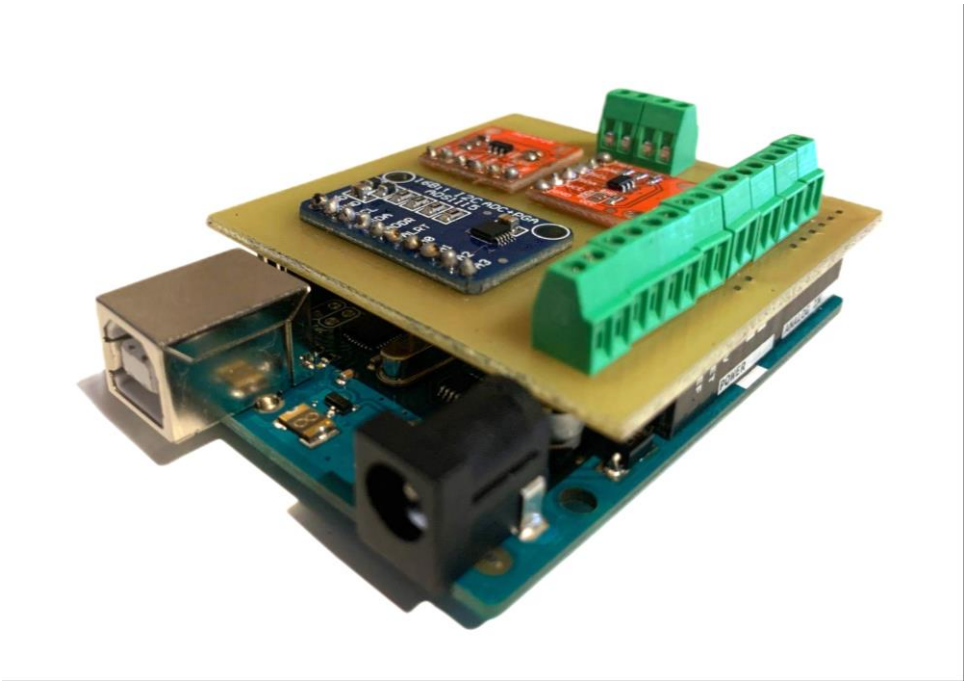


FIGURA 29: CONJUNTO ARDUINO Y PCB

Este conjunto está completo, por lo que se puede introducir dentro de la parte inferior de la caja para proceder al cierre completo del mismo con la parte superior. Las dos figuras inferiores muestran este proceso.

FIGURA 30: PARTE INFERIOR CAJA JUNTO A CONJUNTO PCB-ARDUINO

FIGURA 31: PROTOTIPO DENTRO DE LA CAJA FABRICADA

Con todo ello, el dispositivo físico está completo y listo para instalarse en el computador pertinente para realizar su función. Para un funcionamiento más intuitivo del usuario, a continuación se presenta una figura en la que se marca la nomenclatura de los terminales del prototipo físico con la correspondencia en el servidor OPC expuesta en la tabla 2.

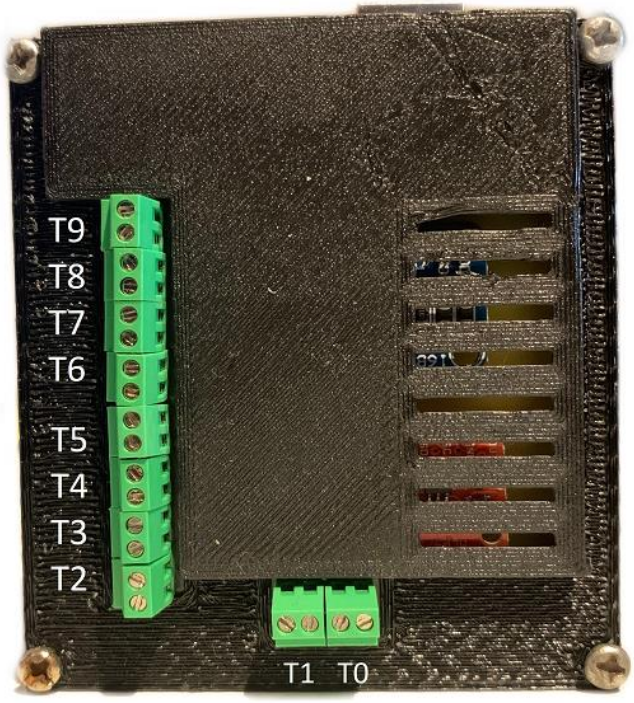


FIGURA 32: NOMENCLATURA TERMINALES FÍSICOS

TERMINALES FÍSICOS	NOMBRE OPC
T0	A.OUT00
T1	A.OUT01
T2	A.IN00
T3	A.IN01
T4	A.IN02
T5	A.IN03
T6	A.IN04+
T7	A.IN05+
T8	A.IN06+
T9	A.IN07+

TABLA 2: CORRESPONDENCIA TERMINALES FÍSICOS CON NOMENCLATURA OPC

5.9 PRESUPUESTO

Englobando todo lo que se ha comentado hasta ahora, se puede hacer un resumen del coste total que supone realizar este prototipo, sin tener en cuenta mano de obra, para a posteriori comparar las prestaciones y precio con el dispositivo TAD que se quiere sustituir. Esto servirá de evaluación para proceder a su sustitución en caso favorable. Como los precios pueden variar dependiendo del momento y lugar de adquisición, para elaborar este presupuesto se han tomado los precios de compra en el momento de la elaboración de este prototipo.

DISPOSITIVO	PRECIO	WEB ADQUISICIÓN
Arduino UNO	24.20€	AMAZON
ADS1115	7.99€	AMAZON
MCP4725	4.5€	AMAZON
PCB	7.4€	PCBONLINE
TERMINALES	1.07€	AMAZON
CONECTORES MACHO	2€	AMAZON

CAJA	~2€	---

TABLA 3: PRESUPUESTO PROTOTIPO

Como se aprecia, el precio total del prototipo está en 48.16€, más de 5 veces menos que el modelo de la compañía 'Measurement Computing' USB-1408FS-Plus que ronda los 275$. En el ámbito económico el prototipo diseñado tiene una cierta ventaja, hay que observar las pruebas y validaciones para comparar prestaciones y poder escoger la mejor opción.

6. INSTALACIÓN

6.1. INSTALACIÓN ARDUINO OPC SERVER

Para la instalación del servidor OPC hay que dirigirse al desarrollador de la aplicación que realiza esta función en 'st4makers.com'. En esta plataforma se puede descargar directamente el software 'OPC Server for Arduino', el cual viene en un archivo comprimido que hay que descomprimir en una carpeta del ordenador donde se quiere alojar el servidor e instalar.

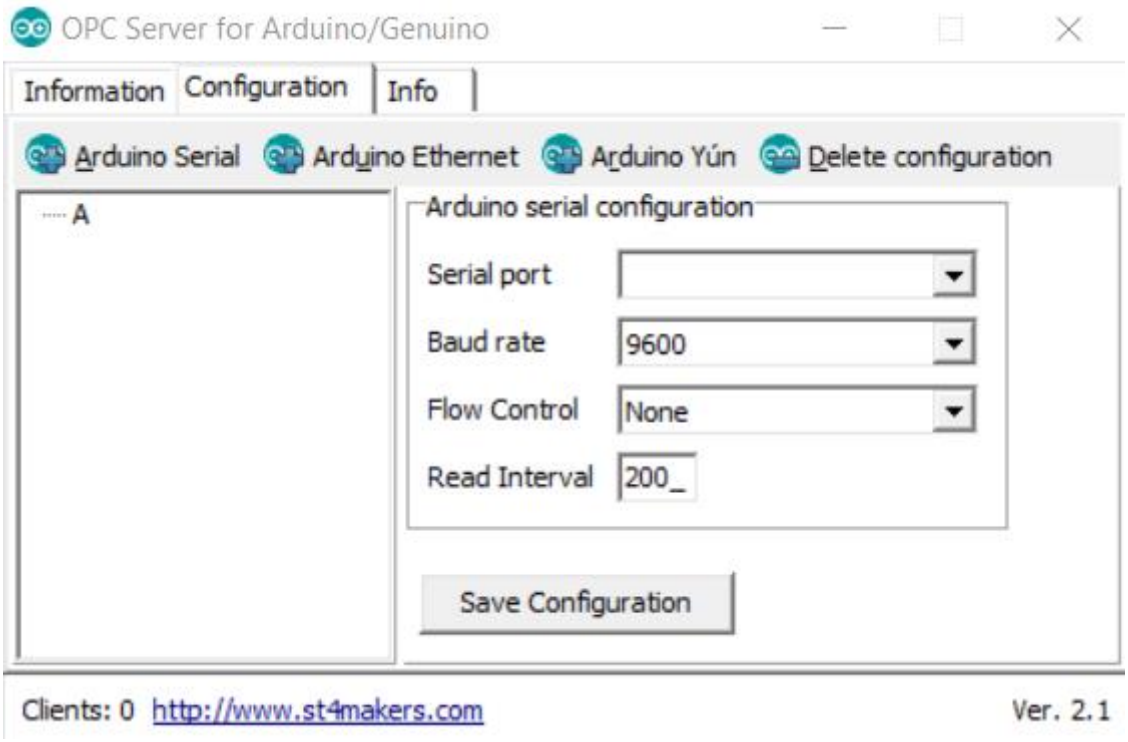


FIGURA 33: PANTALLA PRINCIPAL ARDUINO OPC SERVER

Dentro de la aplicación que viene en el archivo zip, una vez establecida la configuración deseada del servidor a crear, se guarda esta configuración y aparecerá un archivo register.bat para asociar este servidor al computador que se esté utilizando. Este hay que ejecutarlo como administrador para que se lleven a cabo todos los procesos correspondientes y aparezca ya como un servidor al uso. En este caso ya se podría encontrar en la lista de servidores de Softing OPC Toolbox o cualquier otro programa del estilo. Con esto ya estaría instalado el servidor OPC para la plataforma Arduino y se podría acceder a él sin ningún problema con el software de lectura adecuado.

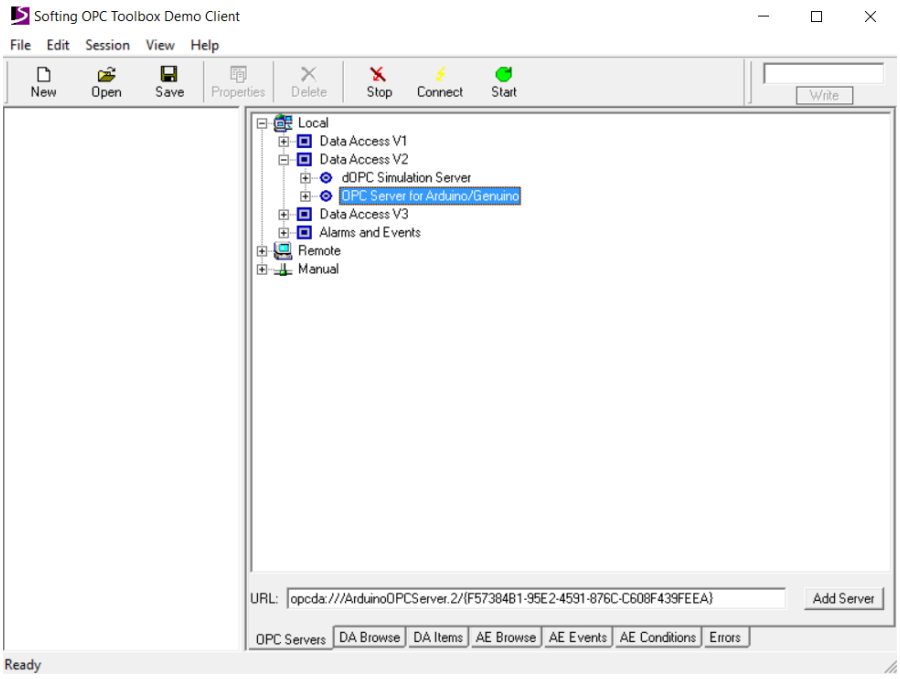


FIGURA 34: PANTALLA SELECCIÓN SERVIDOR SOFTING OPC TOOLBOX

6.2. INSTALACIÓN CONJUNTO ARDUINO

Lo último a instalar en el prototipo es el código de Arduino, el cual debe subirse o almacenarse dentro de la placa conectando ésta con el cable de alimentación al ordenador, estableciendo el puerto de entrada y como se realiza para cualquier otro código, hacer clic sobre el botón subir que verificará la validez del mismo y en caso afirmativo almacenará dentro de la placa en cuestión.

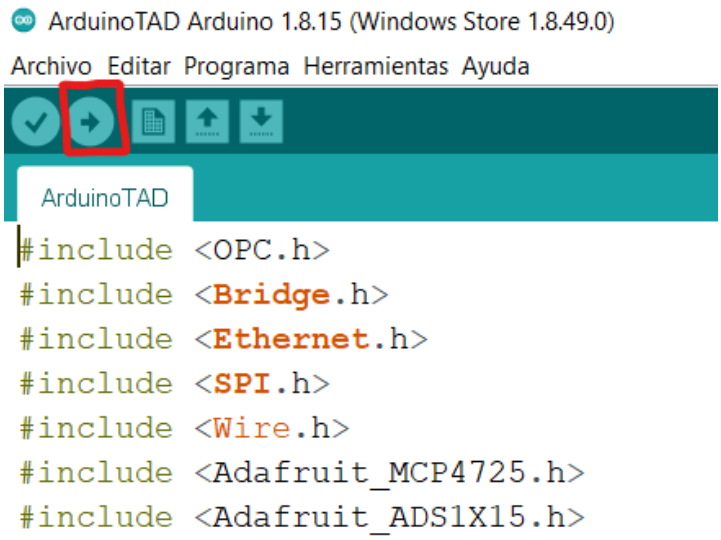


FIGURA 35: BOTÓN SUBIR CÓDIGO PLACA ARDUINO

Una vez dentro de la misma, este código se lee y ejecuta directamente, solo es necesario cargarlo la primera vez y luego alimentar esta placa por este mismo método o directamente a la corriente desde la otra entrada disponible para realizar la función estipulada. Con estos dos pasos todo el conjunto de Arduino está en funcionamiento, solo habría que montar la placa PCB sobre la placa Arduino en los pines correspondientes y el prototipo ya estaría montado al completo, desde cualquier aplicación compatible con un servidor OPC se pueden leer las entradas o establecer voltaje sobre las salidas.

7. VALIDACIÓN Y VERIFICACIÓN

El funcionamiento, por lo tanto, se realiza desde un programa a elegir que tenga la capacidad de conectarse con un servidor OPC instalado en la computadora en funcionamiento. A partir de ahí solo es necesario elegir el servidor OPC for Arduino para encontrar los ítems definidos en el código expuesto en la figura 17. Con un doble click se seleccionan para pasar a una pantalla de control y observación, donde los elementos mostrarán su valor actual y los que sean del tipo escritura estarán habilitados para ordenar un valor concreto para su salida. En el caso de la figura inferior, por ejemplo, se conecta una de las salidas (00) a la entrada 01 para comprobar el valor que se ordena mandar (2 voltios) y el valor que es capaz de leer una de las entradas por defecto.

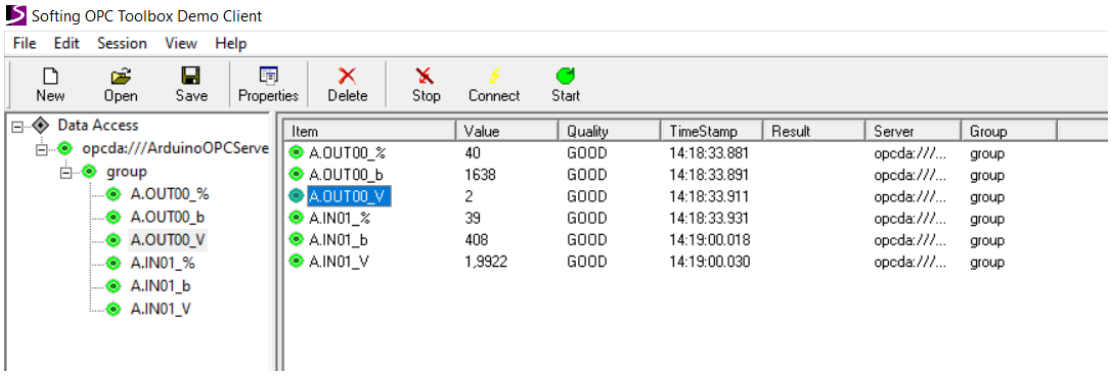


FIGURA 36: MEDICIÓN SALIDA OFRECIDA

Para la validación del prototipo fabricado, lo mejor sería comparar su funcionamiento con el elemento que se quiere sustituir, en este caso la TAD DAQ-USB-1408FS-Plus del laboratorio de control de procesos. Para ello es necesario conectar ambos dispositivos de medición al mismo tiempo a la computadora responsable de su lectura y escritura, para observar en tiempo real las mediciones que llevan a cabo cada uno de ellos. Por otro lado, también se puede comprobar que los valores son correctos con un multímetro común, leyendo la entrada o salida directamente de los terminales de la placa y comparando con lo que se ha ordenado.

Para comparar las salidas con la TAD, se conectan ambos dispositivos a la computadora y se realiza la conexión con sus respectivos servidores. Desde cualquiera de ambos se envía la señal a la bomba para que esta empiece a funcionar y desde ambos dispositivos se obtiene la lectura del sensor de nivel instalado. En el caso de prueba que figura en las imágenes inferiores, se ha enviado una señal a la bomba para que esta funcione al 50% como marca la salida 00 del prototipo, con la nomenclatura ArduinoSerial. Por su parte, la entrada del mismo estándar en ese momento marca 1.1719V correspondiente al sensor de nivel de depósito de la planta en cuestión. La de la TAD por su parte marca 1.1877V en ese momento. La figura 37 corresponde con esta misma prueba instantes posteriores, donde el prototipo lee 1.4795V y la TAD 1.4855V, siempre voltajes superiores pero con pequeñas diferencias provocadas también por diferentes intervalos de lectura, lo que indica

que no siempre marcan la lectura del mismo momento. En rasgos generales, el comportamiento de los dos dispositivos es similar y por lo tanto correcto para una aplicación de estas características, teniendo en cuenta que existe un ruido inherente a las señales de los transmisores por lo que las señales fluctuarán constantemente.

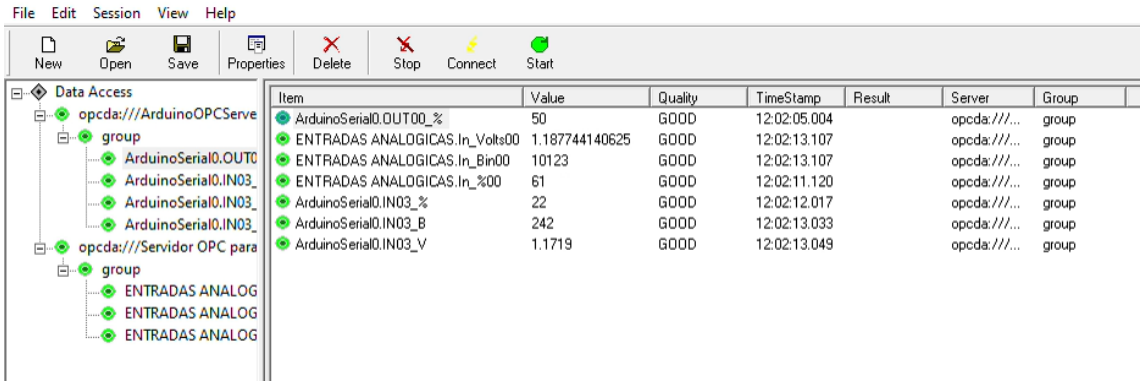


FIGURA 37: COMPARACIÓN 1 ENTRADAS PROTOTIPO FRENTE A TAD

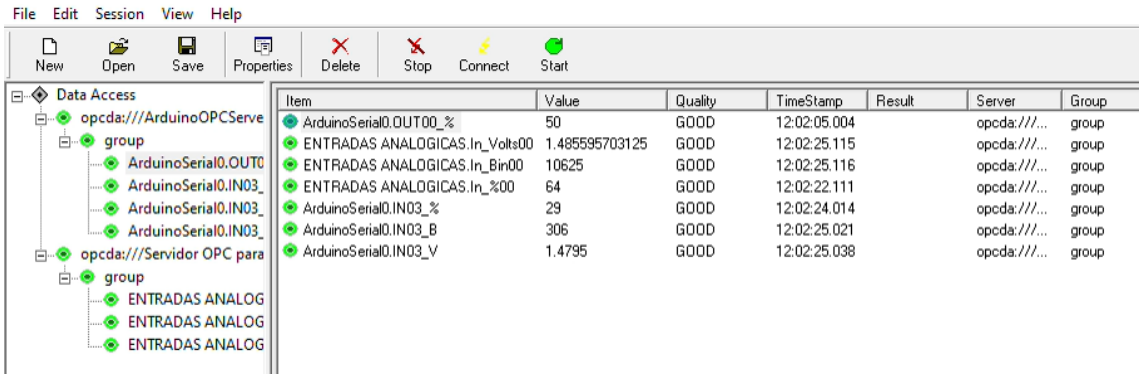


FIGURA 38: COMPARACIÓN 2 ENTRADAS PROTOTIPO FRENTE A TAD

Si se aplica una entrada al actuador con el que el nivel suba linealmente, se puede hacer una barrido de las señales recogidas por parte de los dos dispositivos y compararlas gráficamente. Con esto se tiene una visión del comportamiento general y se puede intuir la diferencia de los valores tomados en los mismos instantes, eliminando parcialmente el error del instante en el que recogen los datos quedando el ruido.

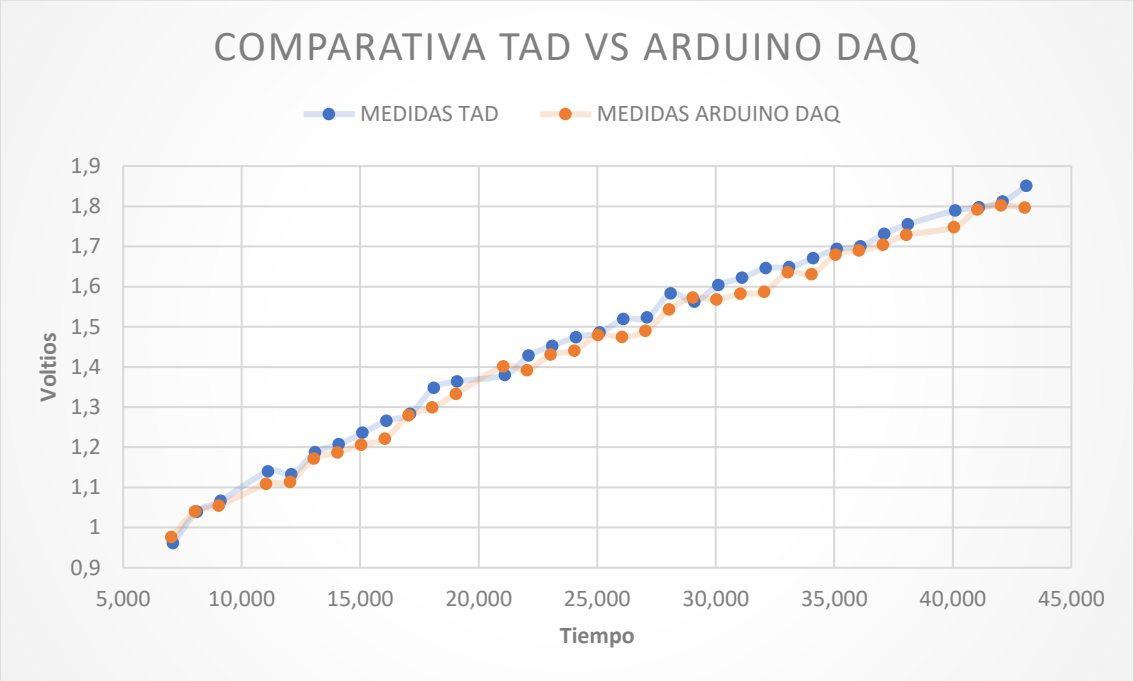


FIGURA 39: COMPARATIVA MEDIDAS ANTE ENTRADA SALTO

Se puede observar un comportamiento parejo en el que las medidas obtenidas desde el prototipo son inferiores a las del TAD en la mayor parte del tiempo pero con un error pequeño. Hay que tener en cuenta que la planta del laboratorio presenta un ruido que se puede observar en su comportamiento en el estacionario, el cual tiene un rizado de 50mV, lo que también se puede considerar como la varianza del ruido, un valor mucho mayor que la diferencia entre estas señales lo cual explica que esta diferencia podría deberse a este fenómeno de alta frecuencia.

Las entradas auxiliares de mayor resolución también se pueden verificar y validar de la misma manera, comparando las lecturas de la TAD frente al prototipo desde una de las entradas del módulo ADS1115 con el mismo experimento anterior, una salida hacia la bomba del 50% de su valor máximo, un nivel del depósito que irá creciendo y un sensor que lo detectará llevando una señal de voltaje al dispositivo de adquisición de datos. En este caso, el prototipo tiene mayor resolución que el propio sistema de adquisición de datos actual y la lectura por lo tanto será más fidedigna que la anterior.

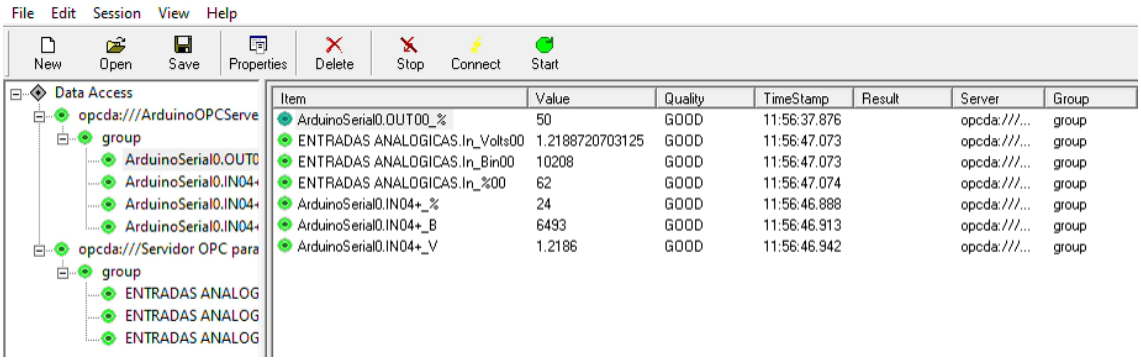


FIGURA 40: COMPARACIÓN 1 ENTRADA PROTOTIPO DE MAYOR RESOLUCIÓN FRENTE A TAD

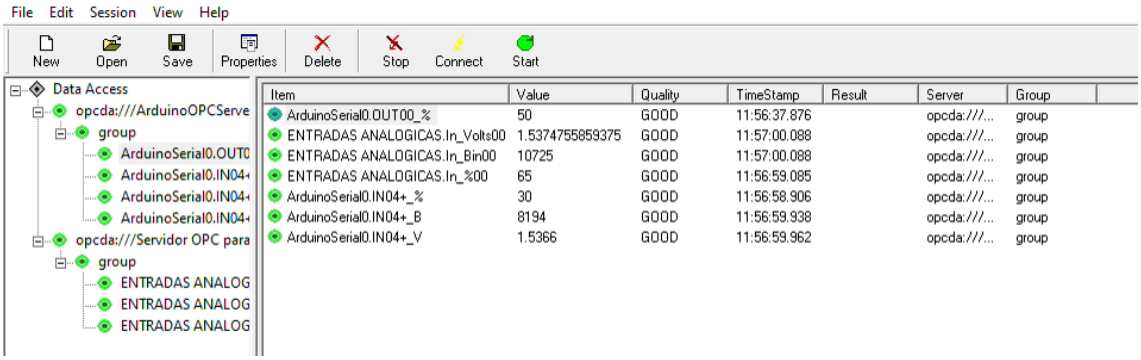


FIGURA 41: COMPARACIÓN 2 ENTRADA PROTOTIPO DE MAYOR RESOLUCIÓN FRENTE A TAD

Como se puede observar en las figuras 38 y 39, las entradas del prototipo en estos dos instantes han sido 1.2186V - 1.5366V y las de la TAD 1.2188V - 1.5374V respectivamente, valores prácticamente idénticos que demuestran la gran capacidad resolutiva de ambos sistemas. Con estos valores y comparando con los de las figuras 36 y 37, se puede también asegurar un buen comportamiento por parte de las entradas de menor resolución características de la placa Arduino para cualquier aplicación al uso que se pueda dar. En el caso de que sea necesaria una mayor precisión en la medida, se tienen estas cuatro entradas con las que obtener la medición, objetivo de la instalación de este módulo ADS1115 en el prototipo.

Realizando el mismo salto que en el caso anterior, aplicando una velocidad determinada a la bomba creciendo el nivel del tanque linealmente, se obtienen las siguientes lecturas en este caso con la entrada de resolución aumentada del prototipo.

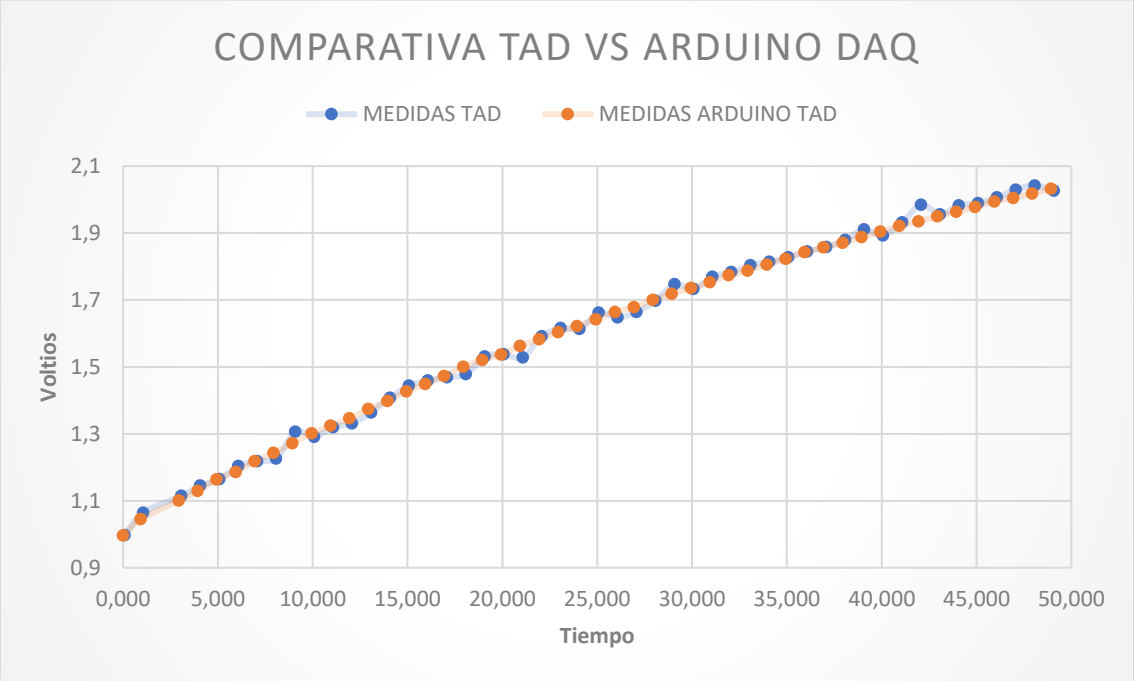


FIGURA 42: COMPARATIVA MEDIDAS PARA ENTRADA + ANTE ENTRADA SALTO

Se observa una gran diferencia con respecto a la entrada básica del prototipo la cual tiene menor resolución; se aumenta la precisión y estabilidad mejorando notablemente el comportamiento general con respecto al dispositivo TAD en uso. Presenta medidas con una progresión uniforme, sin oscilaciones, a medida que el nivel va aumentando.

Con el otro método mencionado, de una manera más simple y común se puede medir las salidas ofrecidas desde el polímetro y la función de voltímetro, observando los voltios que se obtienen respecto a los que se desean. Hay que tener en cuenta que se va a utilizar uno con precisión de un 2% sobre la escala completa en la que se esté midiendo. Para la prueba se introduce una serie de valores, por ejemplo 1.5 voltios y el propio polímetro, desde la escala de 2000m lee 1.513V. Por otra parte, subiendo de escala a 20, con 4 voltios el polímetro detecta 4.03V. A primera vista son valores prácticamente iguales, valores válidos que en principio no puedes reconocer cuales son los más cercanos al valor real pero sí que son valores considerados adecuados.

Con esto se puede deducir, que en el primer caso que se realiza sobre la escala 2000mm los valores medidos pueden variar entre 1.515V y 1.511V, y en el segundo caso que se hace sobre una escala superior, los valores pueden estar entre 4.05V y 4.01V.

Para terminar, se puede hacer esta comprobación con más valores para observar la diferencia entre la medida de las entradas del prototipo, entre las de mayor resolución y menos frente al polímetro.

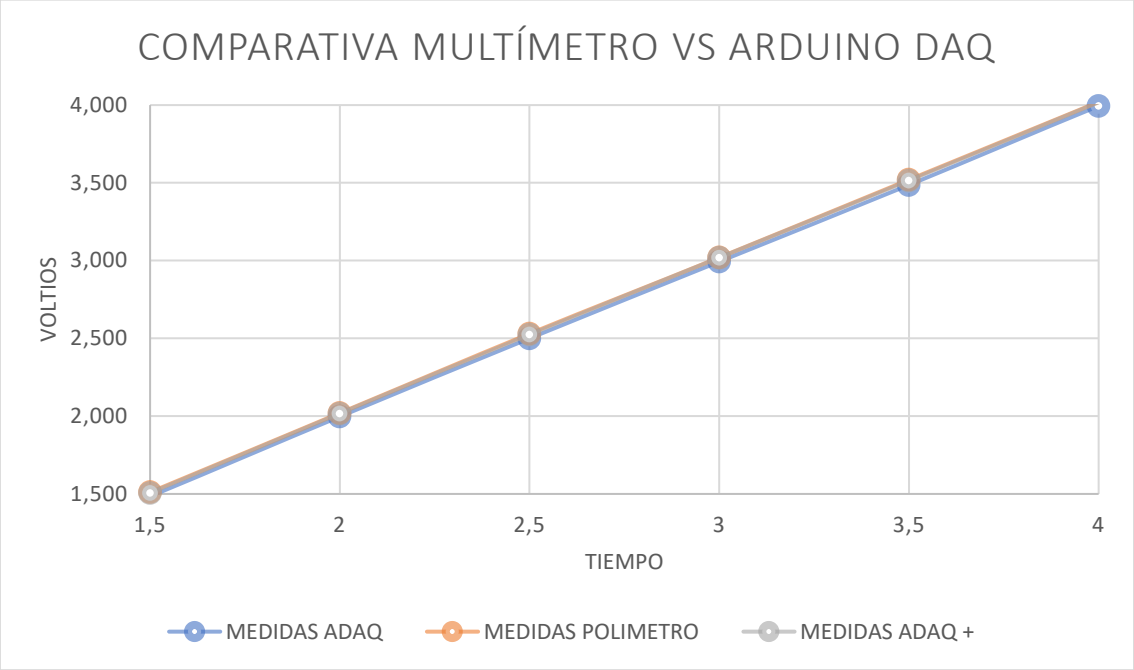


FIGURA 43: COMPARATIVA ENTRE ENTRADAS PROTOTIPO Y POLÍMETRO

La diferencia es casi inapreciable entre la entrada de mayor resolución y el polímetro, la diferencia está en mV. En cambio con la entrada estándar hay mayor diferencia pero sin mayor importancia, siguen siendo valores adecuados para la funcionalidad objetivo.

8. CONCLUSIONES Y POSIBLES MEJORAS

La idea y objetivo desde el primer momento para este Trabajo de Fin de Grado fue la creación de un dispositivo que pudiese hacer las labores de una tarjeta de adquisición de datos a un menor coste y con unas prestaciones similares, con vistas a un futuro poder sustituirla. Para el desarrollo de la misma, se ha pasado por diferentes etapas que han ido marcando la línea a seguir para conseguir cumplir con las expectativas. Es este último apartado de la memoria, se resumen la solución adoptada, se llega a conclusiones sobre la viabilidad y rendimiento del prototipo diseñado, se comentan mejoras o líneas futuras a seguir y se marcan objetivos a largo plazo para conseguir una mejor versión de la misma.

Para el diseño de una nueva tarjeta de adquisición de datos a partir de Arduino, se ha ido pasando por diferentes etapas que han marcado desde la base el prototipo final diseñado. Las primeras dudas a resolver fueron escoger el controlador más adecuado para la función a realizar, teniendo claro desde un primer momento que el protocolo de comunicación más adecuado sería el OPC y por lo tanto que sería necesario un servidor específico para el dispositivo de control elegido. Una vez se tuvo claro la mejor opción posible teniendo en cuenta los módulos adicionales que serían necesarios, solo había que codificar la comunicación y conexión entre todos los dispositivos necesarios. Con el funcionamiento perfeccionado sobre módulos de prueba, ya se podía proceder a mejorar la apariencia exterior y funcionalidad creando una placa de circuito impreso que recogiera todas las conexiones y un envoltorio adaptado al prototipo que lo protegiese de perturbaciones externas. Este resumen de las etapas realizadas hasta llegar al dispositivo de adquisición de datos final ayuda a comprender las decisiones tomadas y los pasos a realizar para mejoras futuras.

A partir de las pruebas realizadas en el apartado 7 de la memoria, se observan unos resultados muy parecidos y similares a los que se obtienen con la tarjeta de adquisición de datos TAD DAQ-USB-1408FS-Plus de la empresa Measurement Computing. Desde este y el prototipo se consiguen lecturas precisas en tiempo real de la instrumentación conectada, permitiendo el control de los actuadores a partir de las salidas analógicas que ambos dispositivos poseen. En cuanto al número de entradas y salidas, contamos con las mismas con características prácticamente idénticas. Ambos dispositivos tienen dos salidas analógicas con 12 bits de resolución y con un rango de 0V a 5V. La mayor diferencia se encuentra en la resolución de las entradas, la cual en la TAD de Measurement Computing se presentan 8 entradas de 14 bits de resolución frente a las 8 que posee el prototipo fabricado, 4 con resolución de 10 bits y 4 con 16 bits, además del impedimento de poder disponer de un modo diferencial a priori; habría que hacer un cambio en el código.

Concluyendo, con la fabricación de este dispositivo de adquisición de datos con Arduino se consiguen unas prestaciones prácticamente idénticas con el modelo comparado; faltaría comprobar cómo sería el funcionamiento en el día a día con una planta real en la que el funcionamiento fuese más largo y exigente. A priori el eslabón más débil y que más fallos ocasiona es el servidor OPC para Ardunio de 'ST4 Makers', el cual es necesario reiniciar cada cierto tiempo y colapsa con relativa facilidad ante

un uso más intensivo. Por lo demás, en estos momentos este prototipo o dispositivo diseñado es una alternativa real para la sustitución de los dispositivos instalados actualmente con un coste mayor, Queda como posible mejora la creación de un servidor OPC para estas placas de Arduino, lo cual por su extensión y dificultad podría llegar a considerarse como otro Trabajo Fin de Grado.

En cuanto a mejoras que se pueden implementar directamente en el prototipo diseñado, con las pruebas realizadas hasta el momento se pueden destacar las siguientes:

- Creación de apoyo físico entre la placa de circuito impreso y la placa Arduino. Los expertos en la fabricación de estas placas comentaron a posteriori que un apoyo al otro extremo de los pines que unen y hacen las conexiones entre estas dos placas mejoraría la estabilidad de la misma impidiendo además la posibilidad de doblar estos pines cuando se separan ambas. En el prototipo existe este apoyo indirecto ya que la separación entre ambas placas es pequeña, por lo que apoyan con facilidad pero sí que es posible doblar los pines de conexión si no se retira con cuidado.

- Instalación de puentes en cada entrada analógica con resistencias pull-down. En el prototipo no han sido instaladas por motivos de espacio y distribución, pero sería una gran mejora para que cuando las entradas analógicas no están conectadas marquen una lectura de 0V y no un valor aleatorio que es el caso ahora. En cuanto existe una conexión la lectura, esta ya es correcta, pero ante pines sueltos al aire la lectura es errónea y puede causar confusión.

El objetivo a futuro es sustituir por completo todas las TAD DAQ-USB-1408FS-Plus del laboratorio por estos prototipos, pero antes de ello se tienen que realizar las pruebas a largo plazo de las mismas para encontrar errores y solucionarlos. En principio es muy posible que estos se instalen a partir del próximo curso y sirvan como una formación más la mejora y adaptación de los mismos.

9. REFERENCIAS

[1] Martínez Marchena, Idelfonso (2013) Arduino OPC Server (Versión 2.1). Windows. Recuperado en marzo de 2021, de:

https://www.st4makers.com/

[2] Manual TAD USB-1408FS-Plus. Recuperado en abril de 2021, de:

https://www.mccdaq.com/PDFs/specs/DS-USB-1208FS-Plus-LS-1408FS-Plus-Series.pdf

[3] Web compañía Measurement Computing fabricante TAD USB-1408FS-Plus. Recuperado en abril de 2021, de:

https://www.mccdaq.com/

[4] Web oficial Arduino. Recuperado en marzo de 2021, de:

https://www.arduino.cc/

[5] Web oficial Espressif. Recuperado en marzo de 2021, de:

https://www.espressif.com/

[6] Comunicación I2C. Recuperado en mayo de 2021, de:

https://www.luisllamas.es/arduino-i2c/

[7] OPC Foundation. Recuperado en abril de 2021, de:

https://opcfoundation.org/

[8] Zamarreño Cosme, Jesús María (2010). Acceso a datos mediante OPC. Santiago de Compostela : Andavira.

[9] Web oficial Adafruit. Recuperado en mayo de 2021, de:

https://www.adafruit.com/

[10] Distribución Pines y especificaciones MCP4725. Recuperado en mayo de 2021, de:

https://www.luisllamas.es/salida-analogica-real-con-arduino-y-dac-de-12bits-mcp4725/

[11] Distribución Pines y especificaciones ADC1115. Recuperado en mayo de 2021, de:

https://www.luisllamas.es/entrada-analogica-adc-de-16-bits-con-arduino-y-ads1115/

[12] Softing AG. Softing OPC Toolbox DA Demo Client (Package V4.10). Windows. Germany. Recuperado en marzo 2021, de:

[13] MicroSim Corporation (1997). DesignLab Design Manager (Version 8.0.). Windows. United States.

[14] Autodesk, Inc (2020). Autodesk Fusion 360 (Version 2.0.10806). Windows. United States. Recuperado en mayo de 2021, de:

https://www.autodesk.com/products/fusion-360/personal

[15] Ultimaker B.V.(2020). Ultimaker Cura (Version 4.9.1.). Windows. Germany. Recuperado en junio 2021, de:

https://ultimaker.com/es/software/ultimaker-cura

APÉNDICE

DATASHEET

ARDUINO DAQ

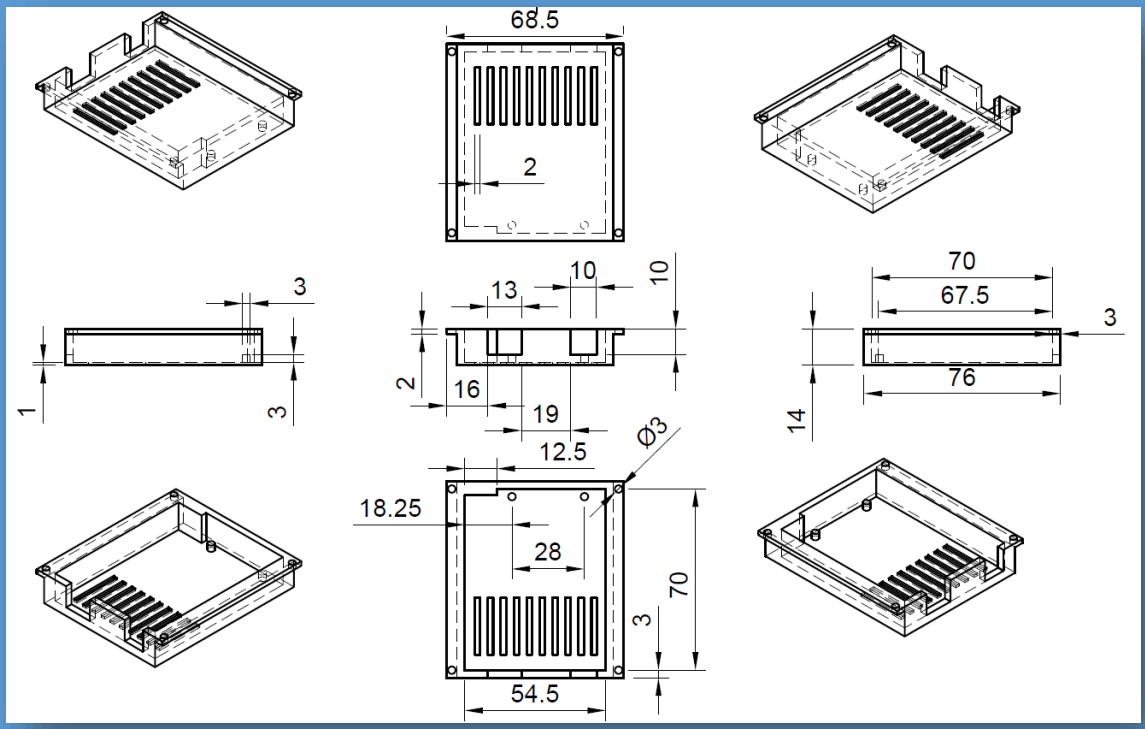
CARACTERÍTICAS PRINCIPALES

- ✓ Comunicación OPC
- ✓ Voltaje funcionamiento 5V
- ✓ 8 ENTRADAS ANALÓGICAS
- ✓ 2 SALIDAS ANALÓGICAS
- ✓ 4 entradas con resolución ampliada 16bits

ESPECIFICACIONES TÉCNICAS

- Entrada corriente recomendada 20mA, máxima 40mA
- 4 entradas 10 bits resolución, restantes 16 bits resolución
- Salidas 12 bits resolución
- Máximo voltaje salida 5V
- Corriente máxima salida 25mA

DIMENSIONES PROTOTIPO



DIMENSIONES PROTOTIPO